

LOAN DOCUMENT

PHOTOGRAPH THIS SHEET

0

INVENTORY

LEVEL

DTIC ACCESSION NUMBER

CECOM-TR-01-9

DOCUMENT IDENTIFICATION

Oct 89

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

DISTRIBUTION STATEMENT

ACCESSION FOR	
NTIS	GRAM <input checked="" type="checkbox"/>
DTIC	TRAC <input type="checkbox"/>
UNANNOUNCED	<input type="checkbox"/>
JUSTIFICATION	
BY	
DISTRIBUTION/	
AVAILABILITY CODES	
DISTRIBUTION	AVAILABILITY AND/OR SPECIAL
A-1	

DISTRIBUTION STAMP

DATE ACCESSIONED

DATE RETURNED

20011101 151

DATE RECEIVED IN DTIC

REGISTERED OR CERTIFIED NUMBER

PHOTOGRAPH THIS SHEET AND RETURN TO DTIC-FDAC

H
A
N
D
L
E

W
I
T
H

C
A
R
E

C2 SYSTEMS DEVELOPMENT BRANCH

CECOM CENTER FOR COMMAND, CONTROL AND
COMMUNICATIONS SYSTEMS

RESEARCH AND DEVELOPMENT TECHNICAL REPORT

***OPERATIONAL PLANNING, PLAN MONITORING AND
THE PHOENIX ARCHITECTURE***

Gerald M. Powell
CECOM

Paul R. Cohen
Department of Computer
and Information Science
University of Massachusetts
Amherst, MA 01003

OCTOBER 1989

CECOM-TR-01-9
Approved for public release.
Distribution is unlimited.

U.S. ARMY COMMUNICATIONS-ELECTRONICS COMMAND
Fort Monmouth, New Jersey

AQU02-02-0195

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

The citation of trade names and names of manufacturers in this report is not to be construed as official Government endorsement or approval of commercial products or services referenced herein.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE October 1989	3. REPORT TYPE AND DATES COVERED Technical Report
4. TITLE AND SUBTITLE OPERATIONAL PLANNING, PLAN MONITORING AND THE PHOENIX ARCHITECTURE			5. FUNDING NUMBERS
6. AUTHOR(S) Gerald M. Powell, CECOM Paul R. Cohen, University of Massachusetts			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) US Army Communications-Electronics Command (CECOM) Software Engineering Center (SEC) ATTN: AMSEL-SE-OP Fort Monmouth, NJ 07703-5207			8. PERFORMING ORGANIZATION REPORT NUMBER CECOM-TR-01-9 (To DTIC: October 2001)
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSORING / MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE
13. ABSTRACT (Maximum 200 words) The principal topic of this report is the dynamic behavior of military organizations. In particular we address issues associated with characterizing the ebb and flow of battle such as how to assess progress being made by our forces. The report describes a representation mechanism for monitoring plans as they are executed which supports communication between plan elements as well as plan modification in light of revisions to objectives and changing battlefield conditions. The overall planning model presented here, within which the plan-execution monitoring capability is embedded, is perhaps the first system in the artificial intelligence literature to achieve real-time planning and monitoring, and it will soon be capable of real-time planning during execution. The cornerstone of the model is the concept of envelopes which are data structures designed to measure progress and facilitate communication within an organization in dynamic environments.			
14. SUBJECT TERMS Command and control; planning; operational planning; execution monitoring; plan monitoring; replanning; envelopes; artificial intelligence; real time; Phoenix; modeling and simulation; military planning; plan execution monitoring; battlefield planning; knowledge based systems; expert systems			15. NUMBER OF PAGES 79
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

Contents

1	Introduction	3
1.1	A Summary of the Sections in this Report	4
2	The Operational Planning Problem	6
2.1	Operational Planning	6
2.2	Beyond Doctrine	13
3	The Relevance of AI Planning and Problem-solving Techniques	15
3.1	Projective Planning	15
3.2	Reactive Planning	16
3.3	Probabilistic Planning	17
3.4	Local Search	18
4	The Fire Problem.	18
5	The PHOENIX Architecture	21
5.1	Implementation of PHOENIX	36
6	Improvements to the PHOENIX Architecture	42
6.1	Envelopes	43
6.2	The Utility of Envelopes.	44
6.3	How Envelopes Integrate Projection and Reaction.	45
6.4	Agent Envelopes and Plan Envelopes.	47
7	Envelopes in Operational Planning	49
7.1	Agent Envelopes and Multi-agent Plan Envelopes	51

8 Mapping the PHOENIX Architecture, with Envelopes, to the Operational Planning Problem	55
9 Conclusion	59

1 Introduction

The principle topic of this report is the dynamic behavior of military organizations. What happens once an engagement begins? How can we characterize the ebb and flow of battle? How can we characterize the progress of our forces, viewed at any level of granularity, from platoons all the way up to divisions? What messages need to be passed between echelons to ensure coherent behavior across spatial and temporal extents?

Although our intention was originally to study the static aspect of the operational planning problem, in which resources are allocated in anticipation of engagements, it quickly became apparent that military planners need representations and inference mechanisms to deal with dynamic situations as they evolve. That is, in addition to constructing plans for various alternative scenarios, we need mechanisms to monitor plans as they are executed, to maintain our aims under widely varying circumstances, and to modify our plans and goals as appropriate.

Until recently, AI planning research emphasized the generation of plans. Little attention was paid to the processes of executing and monitoring plans once they were developed (Sec. ??). The locus of intelligence in AI planning techniques was plan-generation, not plan execution, monitoring, and replanning. The model presented here is perhaps the first system in the AI literature to achieve real-time planning and monitoring, and it will soon be capable of real-time replanning during execution. Thus, it became the focus of the research reported here. Although we don't have protocol data to support the model we develop below, we can justify the effort of developing the model in four ways:

- The model provides a framework for focusing knowledge acquisition, for interpreting the behavior of planners as they perform plan monitoring and replanning tasks, and for generating hypotheses in an effort to identify those aspects of the tasks that are difficult for human planners. Thus, it provides a starting point for the design and development of automated aids to monitoring and replanning.
- We need the model because common sense tells us that plans never evolve the way they are supposed to, so planning mechanisms that lack plan-monitoring/plan-modification mechanisms are incomplete and of limited utility.
- We need the model because it tells us important things about the nature of communications between and within echelons.
- The model makes predictions about dynamic behavior in military organizations. Thus, it should be further developed and tested. Any predictive mechanism can be applied not only to our own forces, but also to adversarial forces, and is therefore potentially useful.

The cornerstone of the model is the concept of *envelopes*. Envelopes are data structures designed to measure progress and facilitate communication within an organization in dynamic

environments. We will report on a planning architecture, called PHOENIX, for dynamic, real-time planning with envelopes.

Although PHOENIX is designed for real-time control of forest fires, its task is similar in many ways to that of the operational planner: resources must be allocated to control a threat, and those allocations must be reinforced and otherwise dynamically adjusted in real time (see Sec. 8 for more detail). PHOENIX's *environment* is also similar in many ways to that of the operational planner: both are spatio-temporal environments; terrain and weather are important, but weather, at least, is uncertain; the threat behaves differently in different terrains and ground covers; there can be multiple threats, distributed in space and time; and the resources at the planner's disposal have different capabilities. The major differences are that forest fires are not as complex as the European Theater, and forest fires are not sentient adversaries.

One can read this report in several ways. It is, at minimum, a discussion of the relevance of an advanced AI architecture to the operational planning problem. We hope it will also be seen as a partial design for an AI operational planning system. How would such a system be used? Perhaps the most stringent test, short of actual use in the European Theater, would be to implement an operational planner's assistant for use in war games at the War College. The system would be an assistant, rather than an autonomous planner, because the responsibility for plans must rest with human planners, and because the planner's assistant will itself require assistance with tasks, such as perception, at which computers are historically poor. One can imagine applying AI techniques to many aspects of the operational planning problem, but our assumption in this report is that a planner's assistant is the most challenging problem and also the one with the biggest payoff.

1.1 A Summary of the Sections in this Report

Several sections comprise this report:

The operational planning problem. (Sec. 2.) We will discuss two aspects of the problem, which we call *static* and *dynamic*, respectively. The static problem involves allocating resources prior to engagement, whereas the dynamic problem involves monitoring and managing resources during engagement.

The relevance of AI planning and problem-solving techniques. (Sec. 3.) Here, we survey the AI literature for techniques that pertain to aspects of the operational planning problem. Specifically, we will discuss the two major classes of planning techniques—hierarchical and reactive planning—and some recent work that suggests local search (an operations research technique) may be very useful to both the static and dynamic aspects of the operational planning problem. The emphasis of this section is on the strengths and weaknesses of many techniques, since we believe the operational planning problem will require many such techniques, used to their best advantage.

The fire problem. (Sec. 4.) We will discuss the fire-fighting problem to set the stage both for the following discussion of the PHOENIX architecture (Sec. 5), and for the comparison between the fire-fighting problem and the operational planning problem (Sec. 8).

The PHOENIX architecture. (Sec. 5.) PHOENIX is an architecture for controlling simulated forest fires. We will describe its motivation and design, specifically its mechanism for integrating reactive and projective planning. We will also illustrate in detail those parts of PHOENIX that are currently operational.

Improvements in the phoenix architecture. (Secs. 6 and 7.) Recently, we have developed an idea that ought to streamline the interaction between planners and the agents that carry out the plans (e.g., between commanders at different levels of the military hierarchy). *Envelopes* are metric spaces in which agents monitor their progress. The idea is to limit the interactions between planners and their agents to those places where progress is unexpectedly good or poor, and for the planner to use the envelopes of its agents to project their progress in the future. Envelopes are not yet implemented in PHOENIX but are so clearly relevant to the operational planning problem that we will offer several examples.

Mapping the PHOENIX architecture to the operational planning problem. (Sec. 8.) In this section, we will formalize the intuitions that PHOENIX is a good architecture for operational planning. We will look critically at the similarities and differences between the fire fighting and operational planning tasks, and also at the structure of the planners for accomplishing these tasks.

2 The Operational Planning Problem

This section discusses the operational planning problem as understood in doctrine, and extends that understanding in two areas: COA generation based on our observations of planning activities of experienced Corps planners, and plan monitoring and replanning as modelled in an AI planning domain that appears to share many of the characteristics of the operational planning problem.

2.1 Operational Planning

A U. S. Army corps headquarters is responsible for the conduct of war at the operational level. The operational level of war is the link between strategy and tactics, characterized by "the repositioning or displacement of large units ... to force maximum strength against the enemy's weakest point thereby gaining strategic advantage."¹ To contrast the operational and tactical levels of warfare, operational success is gained through establishing positional leverage on the battlefield, forcing the enemy to change dispositions and plans to meet unexpected threats, and to cause indecisiveness within the enemy command and control structure². Tactical success is gained by massing decisive firepower at the critical point and time on the battlefield³. Operational planning is a difficult, cognitive and visual task, currently unaided by automated systems. This section provides a description of operational planning as specified both by doctrine and by experienced corps planners. It identifies the information requirements of planners, the major decision elements within a plan, and the doctrinal process planners are taught.

Planning at the corps level is a continuous process. At any time, the commander's planning staff will be operating in one of two contexts: planning for a new operation as directed by higher headquarters, or planning operations to react to the changing battlefield situation. In either context, the staff must project battlefield situations into the future (approximately 72 hours at the corps level) and plan based on that projection. A corps plan will encompass actions involving up to 150,000 soldiers and extending for periods longer than a week. With modifications to the plan (Fragmentary Orders or FRAGORDS) the life of a corps plan could be indefinite. In spatial dimensions the corps plan will govern operations over a space at least 150 kilometers wide by 150 kilometers in depth, but this can also vary greatly due to variances in terrain, expected enemy resistance, and the type of operation the corps is to conduct. Additionally, the corps plan will address operations directed at an area beyond their forward boundary, the extent of which is defined by movement times^{4,5}.

¹U. S. Army TRADOC, Airland Battle 2000, August 1982.

²COL E. White, "Deep Battle Operational Concept", Deep Battle Planning Briefing, Ft Leavenworth, KS, June 28, 1985.

³COL H. Adams, COL J. Allen, COL R. Herrick, COL V. Letonoff and COL M. Moran, Planning Discussions held at U. S. Army War College, Carlisle Barracks PA, 30 July 1985.

⁴HQ, Department of the Army, FM 100-5, Operations, 1982.

⁵The Area of Influence is the area within 72 hours of the corps FLOT (Forward Line of Own Troops),

Planning commences upon receipt of an operations order or operations plan from the immediately superior headquarters. The conceptual components of this order include a situation description and a corps mission. The situation description designates terrain over which the corps is to operate, the forces over which corps has control, and a description of the enemy forces the corps will operate against. In developing a plan or order the corps commander and staff will make a number of decisions addressing a set of key plan issues. These decisions will then focus the remainder of the staff activity in developing the complete plan. Planning terminates when the commander approves the operations plan or order. An operations order is an order issued by a commander specifying missions and tasks to be accomplished by his subordinate units. The order normally specifies a time at which the operation is to commence, and, unless superseded by a later order, subordinates are to take all necessary action to begin execution at that time. An operations plan is prepared to be executed at some future time, normally unspecified. Operations plans are normally prepared in advance for certain contingencies, and are executed upon order of the commander.

The situation description and mission in some sense correspond to the initial state and goal state concepts found in many AI planning theories. The decision-making issues correspond less directly to the operators available in the problem domain. The complexity of the operational planning problem becomes evident when we characterize the initial and goal state descriptions, and the relationship of the operators to these initial and goal state descriptions.

Situation Description. Military plans are to be executed in a battlefield situation, and are intended to change particular aspects of that situation. Battlefield situations are difficult to understand, however, and describing those factors relevant to the current planning task involves the continuous effort of many individuals. The complexity of battlefield situations is best described by examining each of the relevant factors and characterizing what is known about the factor against what needs to be known.⁶

Enemy Forces. The enemy force opposing the projected corps operation is a critical element of the battlefield situation. The corps has extensive intelligence collection resources available for obtaining information about the enemy, and maintains a database of the enemy force (enemy order of battle, EOB) at all times. This EOB includes everything known or suspected about the enemy force, such as units identified, locations, activities, strengths and command relationships. While the EOB does provide a snapshot description of the enemy at any time (see sections Situation Analysis and Situation Analysis (SA) Actor for further description), it does not provide the description of enemy forces required by planners.

The information required by planners describes what the enemy is likely to do (intentions), and what the enemy can do (capabilities) to react to corps operations. This information must

an imaginary line connecting the most forward positions of corps elements. The corps will conduct fire and intelligence operations into the Area of Influence. The Area of Interest is the area within 96 hours of the FLOT. The corps will conduct intelligence operations in the Area of Interest.

⁶These factors are derived from the METT acronym found in numerous doctrinal publications (e.g., see HQ, Department of the Army, FM 71-100, Armored and Mechanized Division Operations, 1979.) which stands for Mission, Enemy, Terrain, and Troops available.

be derived from the description provided by the EOB. This derivation process is one measure of the complexity of the enemy forces segment of the situation description. Other measures focus on the quality of the information contained in the EOB. This information is often inaccurate, incomplete, redundant and out-of-date.

Terrain. A second critical element of the battlefield situation is the terrain over which the operation is to be conducted. As with enemy forces, a database of terrain information is maintained by the corps as a set of terrain factor overlays.⁷ These overlays provide information about terrain factors, such as vehicle movement rates, areas which constitute obstacles to movement, and water obstacle crossing sites (bridges, bridgable sites, and fordable sites, for example). As with the enemy forces situation, this terrain database does not provide the description of terrain required by planners. Planners need to know the critical terrain features (key terrain, avenues of approach, etc.) and what their impact will be on combat operations. In contrast to the enemy situation, however, the information contained in the database is highly reliable and provides an excellent basis for developing the required terrain description.

Friendly Forces. Friendly forces provide the third element in the battlefield situation description. Of the three, the information available for friendly forces most closely resembles the description required for planning purposes. The friendly forces database contains information similar to the EOB information on enemy forces. Unlike the enemy forces situation, however, only a more general form of unit capability is required for planning, and this can be quickly estimated based on unit type and strength information. Also, the information quality is high, reducing the other source of complexity.

Mission. The mission, which is the input that initiates the planning problem, is a description of what the corps is to achieve (see Receive Mission, Figure 1). It is usually described as a task or set of tasks for the corps to accomplish. Additionally, the mission may contain certain constraining information, such as absolute (secure Objective Charlie no later than), or relative (secure Objective Bravo, then secure objective Charlie) times, or contingency tasks (be prepared to counterattack).

The mission given the corps is incomplete in that it describes only a final, or top-level set of tasks to accomplish. Accomplishment of this final set of tasks may entail accomplishment of a number of other tasks. The complexity of the mission statement in the operational planning problem is a result of this incomplete mission description and the resulting necessity to perform considerable refinement and/or specification of it before devising a plan.

Key Decision Issues. In devising a plan to accomplish a mission the corps planner must address a set of decision issues. These issues constitute the major plan elements, and collectively describe the concept of the operation. These issues also reflect the means (plan elements) a corps commander has available to effect the required changes in the battlefield situation necessary to accomplish the mission. The relationship between the situation and

⁷These are termed overlays because they consist of symbology written on clear acetate sheets keyed to specific areas on topographic map sheets. When overlaid on the map they provide condensed information about specific terrain factors in the area represented on the map.

these plan elements is indirect, in that the elements do not modify the battlefield situation. Instead, they force a decomposition of the corps planning problem into a set of simpler (e.g., division, armored cavalry regiment, corps artillery), planning problems, which are then pursued independently. Means of controlling the interaction between these subproblems are reflected in the corps plan's scheme of maneuver (section Scheme of Maneuver) and control measures (section Control Measures) elements. The corps commander and staff retain the function of composing these subproblem solutions as they evolve during the execution phase of the battle.⁸

At the corps level, the basic operator set can be classified into three categories: resource allocation operators, subgoal allocation operators, and, to a lesser degree, control operators.⁹ The resource allocation operators create the entities and situation description (terrain allocation, friendly forces available, implicit allocation of enemy forces, etc.) for the subproblems, the subgoal allocation operators create the goals, or missions, for the subproblems, and the control operators impose control over the subproblems as subproblem solutions occur. These plan operators are embedded in the five plan elements discussed in the remainder of this section.

Scheme of Maneuver. The scheme of maneuver describes and functionally relates the activities of all subordinate elements of the corps, and describes the relationship of these activities to the overall corps mission. The scheme of maneuver is the unifying element of the corps plan. Doctrinally, all activities of the corps are conducted in support of the scheme of maneuver¹⁰.

Task Organization. The task organization allocates the maneuver resources¹¹ of the corps to subordinate command and control entities. The task organization describes the subordinate elements of the corps addressed in the scheme of maneuver in terms of resources they will have available for the operation. It is interesting to note that the number of subordinate elements is not fixed, and that the corps planners have a degree of flexibility in establishing the number of subordinates required.

Mission Allocation. The mission allocation issue refers to the identification of specified (those specifically stated by higher authority) and implied (those identified by the corps planner as necessary to accomplish the specified missions) missions, and the assignment of these missions to the subordinate elements identified in the task organization. Once the specified and implied missions are identified by the planner, they all become specified missions for the subordinate elements.

⁸In military terminology, this problem of composing subproblem solutions is referred to as operations control. Other terms familiar to the planning research community are plan execution, plan monitoring, and plan revision.

⁹Control operators allow the planner to express coordination required between subunits, or subproblems. Examples of control operators are objectives, sequencing control measures and specific actions.

¹⁰T. T. Bean, M. A. Ottenberg and R. K. Mukherjee, Command Control Subordinate System Functional Analysis: Maneuver Control Functional Segment, Technical Report MTR-83W00022, MITRE Corporation, McLean VA, April 1983

¹¹Maneuver resources are those units which engage the enemy forces in direct combat and seize and occupy terrain. At the corps level, the maneuver resources available are the armored and infantry divisions, their subordinate maneuver resources (brigades), and the cavalry regiment.

Control Measures. The control measures issue can be viewed as another resource allocation issue in which terrain features and areas are allocated to the subordinate elements identified by the task organization. Control measures can also be used to specify a temporal ordering on the actions of subordinate elements when this is required. At this point in our research it is not clear to what extent temporal orderings of actions are important to corps operations. Control measures are normally portrayed in a graphic manner, with an operations overlay being the standard technique for display.

Support Priorities. Support priorities can be viewed as another mechanism for allocating resources, the resources in this case being functionally specialized, such as artillery and engineers. These resources will be allocated to subordinate elements on a priority basis, with the planner specifying a priority ordering for supported units, which must identify at least the highest priority, and a functional type of support being provided.

The Problem-Solving Approach. In attempting to solve a particular planning problem, military practitioners are taught to follow a decision-making process defined by doctrine¹². In general terms, the process attempts to fully specify information relevant to the current problem, identify a number of alternative, incompletely specified plans, or solutions, then evaluate these alternatives and complete the selected plan. Figure 1, The Military Planning Process, which has been adapted from [8], illustrates the major activities in this process. In Figure 1, the squared rectangles represent staff actions and the rounded rectangles represent commander's decisions.

The process begins when a new mission is received. The staff immediately begins analyses of their respective functional areas¹³ as they pertain to the unit's new mission. The commander, with the aid of his principal operations assistants, conducts a mission analysis, which will produce a complete refinement and statement of the corps mission. Based on this new mission statement, the staff begins producing and exchanging the information required for developing plan alternatives. This information is integrated in two concurrent processes, the courses of action and enemy capabilities processes. The output of this integration effort is a set of distinct partial plans (courses of action), each of which could serve as the basis for a complete plan. These candidates are evaluated by the commander and staff, and a single alternative is selected as the concept of the operation. The details of this evaluation process are outside the scope of this research. The commander may then amplify this concept with commander's guidance as he sees fit. This concept of the operation and the commander's guidance will then drive the remainder of the planning activity as the staff develops the complete plan. In examining this planning model it is apparent that a number of these processes are information-producers and a number are decision-producers¹⁴. The issues being investigated in this research are the decision issues, plus those information issues which are too closely intertwined with the decision issues to isolate as information products. Based on our preliminary domain investigation the key

¹²HQ, Department of the Army, FM 101-5, Staff Organization and Operations, 1984.

¹³The major staff functional areas at corps headquarters are as follows: G1, personnel and administration; G2, intelligence, weather and terrain; G3, operations; and G4, logistics.

¹⁴Information-producing processes produce, as output, information which is used by other processes. Decision-producers produce, as output, elements of the plan.

decision and information processes that require investigation can be described as the following set of tasks.

Mission Analysis. The purpose of the mission analysis process is to clarify, understand and restate¹⁵ the mission given the corps in the higher command's operations plan or order. This process is necessary for multiple reasons. The most important of these is that the higher planning staff has developed its plan, and corps mission statement, without a detailed analysis of the characteristics of the battlefield environment relevant to the corps. These characteristics will influence how the corps can accomplish its given mission, and will probably dictate other tasks which must be performed. A second reason is that the corps plan to accomplish this mission must be consistent with the superior's concept of the operation, and this requirement may limit the flexibility of the corps planning staff in developing their plan. In performing a mission analysis the commander and staff are attempting to determine three sets of requirements¹⁶. The elements of the first set are those tasks to be performed. This includes not only the tasks specified in the higher command's order, but also those tasks not specified but which are essential if the specified tasks are to be accomplished.¹⁷ For example, securing an objective (specified task) may require moving across a large water obstacle such as a river. The corps commander may feel that securing a water crossing site (implied objective) is essential to accomplishing this objective, and therefore make this a task to be accomplished. The second set identifies the purpose (relationship to superior command's mission) of each task assigned the corps, and is principally used to understand the superior command's concept of the operation and how the corps operation fits into this concept. The final set identifies constraints on the corps' actions which the planning staff needs to consider. These constraints may be specified by the superior command or developed by the corps planners.

These constraints may be temporal (accomplish a task no later than, by a specified time, etc.), spatial (do not cross international boundaries, do not withdraw beyond a specified line, etc.) or operational (no air support, maintain unit integrity, etc.). The output of the mission analysis process is a restated mission, which is a statement of all tasks the corps is to accomplish and their relationship to the corps' mission. It includes only those tasks the commander feels are essential to accomplish the mission assigned the corps in the higher operations plan or order. If multiple tasks have been identified, then any required ordering of these tasks is also identified. It is important to note that routine tasks, and tasks normally included in the scope of command are not part of the restated mission¹⁸. When complete, the restated mission provides focus for the remainder of the planning effort.

Terrain Analysis. "Land battle takes place amid natural and man-made variations in terrain. How best to accomplish the mission is frequently resolved in terms of this terrain." [6] Included in the terrain and enemy estimates process of Figure 1 is a detailed terrain analysis

¹⁵Op. Cit. and U. S. Army War College, Sound Military Decision Making, Study Supplement, Carlisle Barracks PA, September 1982.

¹⁶Op. Cit.

¹⁷These are referred to as specified and implied tasks. At the conclusion of the mission analysis phase they will all become specified tasks.

¹⁸Op. Cit.

effort which attempts to completely describe the militarily significant aspects of the terrain over which the corps is to operate. One examination of this process ¹⁹ concluded that a typical analysis for a corps operation would require seventy five map sheets and produce 814 terrain factor overlays. Clearly, the key decision-makers cannot be concerned with this level of analysis. The terrain effort described here assumes the products of this detailed terrain analysis as being an available information product. The intent, then, is to identify those characteristics of the terrain which are likely to have a decisive influence on the conduct of the operation [9]. This will allow the planner to identify those application features²⁰, such as avenues of approach and obstacles, which will impact the operation, and also understand the impact of each feature.

Situation Analysis. Situation analysis is an important element of battlefield planning for several reasons. First, because they cannot exert control over enemy forces, planners want to know as much as possible about their disposition and activities. From this information planners derive estimates on enemy capabilities and intentions and incorporate these beliefs in their plan. Second, planners want to be informed on the enemy situation because the enemy pursues objectives deleterious to corps forces and accomplishment of the mission. Third, corps missions are often targeted at enemy forces. The goal of the situation analysis process is to identify and evaluate enemy capabilities, and to infer enemy intentions. Capabilities are viewed in terms of the potential of the enemy to react to friendly forces actions whereas intentions are not necessarily dependent on friendly forces actions. Enemy intentions are viewed as less reactive than enemy capabilities. They are deliberate objectives which the enemy seeks to achieve by some relatively well-developed plan. Doctrinally, it is preferred to focus on enemy capabilities and to resort to inferring enemy intentions only when capability analyses are insufficient ²¹.

Friendly Force Analysis. For planning purposes, the friendly force analysis process (friendly estimate, logistics estimate in Figure 1) is characterized by the presence of great quantities of hard, reliable data. These data describe the available units with both objective (type, strength, equipment status, ammunition and fuel status, etc.) and subjective (readiness, morale, training, etc.) attributes. The purpose of the process is to determine relative capabilities of the resources the planner has available.

Course of Action Generation. The course of action generation process develops alternative, incompletely specified plans referred to as courses of action.

Key Decision Issues. The course of action process integrates the information provided by the staff estimates processes of Figure 1 into a set of distinct alternatives, each of which will accomplish the mission determined by the mission analysis process. As indicated in Figure

¹⁹E. Trelinske, Proceedings of the Symposium on Computer Graphics in Support of Tactical Command and Control, TRADOC Combined Arms Test Activity, Ft. Hood TX, August 1977.

²⁰M. Gronberg, Terrain Issues Study, Technical Report CR-0003, CENTACS, CECOM, Ft. Monmouth, NJ, May 1985.

D. G. Shapiro, P. S. Marks and J. M. Abram, Battlefield Commanders Assistant, Technical Report TM-1058-01, Advanced Information and Decision Systems, Mountain View CA, 1985

²¹U. S. Army War College, Simulated Planning Exercise, Carlisle Barracks PA, 1984.

1, the interaction between course of action generation and enemy situation analysis should be extensive.

Plan Monitoring and Replanning. After the selected course of action is developed into a complete plan, the plan must be disseminated to subordinates. The corps must continually assess the battlefield to estimate the impact of the changing situation on the disseminated plan. Once the plan begins execution, certain actions of the plan as well as other conditions of the battlefield situation must be monitored to determine if the corps objectives are being achieved in accordance with the plan. Battlefield events may result in the need for the corps to replan. These events may be of the type that indicate the plan may fail or of the type that indicate an unexpected opportunity for success (Note: If the corps has sufficient time, it develops contingency plans for those battlefield situations it believes may develop and which could require a significant change in the plan). Battlefield planning thus seems to require a knowledge of how to identify conditions that need to be monitored with respect to a plan whether it is prior to, or during, plan execution. Some of this knowledge may be situation independent whereas other of it will depend significantly on such factors as the particular area of operations, the particular friendly and opposing forces, and the particular mission.

2.2 Beyond Doctrine

Further formulation of the planning problem. Powell and Schmidt analyzed the planning activities of Corps-level human planners. This study arrived at the following conclusions:

- Corps planners have a significant requirement to further formulate the planning problem given to them. This further formulation is an interpretive and predictive task directed at both the initial state (terrain, weather, friendly and enemy forces) and the goal state (mission and commander's guidance) of the planning problems they receive.
- These problem formulation activities are interleaved with other planning tasks, specifically, problem decomposition, plan evaluation, and plan repair.

Monitoring and Replanning: Although we lack empirical data on plan monitoring and replanning in human planners, we have derived the following characteristics of these aspects of planning from ST 100-9 [9] (*The Command Estimate*) and FM 101-5 [30] (*Staff Organization and Operations*):

Planning is Opportunistic Battlefield events may indicate unanticipated opportunities for success, which should be exploited for retaining or regaining the initiative (see Appendix B, [9]).

Planning for Contingencies If the Corps has time, it "continuously develops contingency plans for new alternatives to current operations." (see Appendix B, [st101- 9?]).

In addition to the few characterizations of planning that we can derive from the available literature, we make the following assumptions about human real-time planning in dynamic, real-time, spatially-distributed, multi-actor environments such as the European Theater:

Planning is Incremental. Execution begins before the plan is fully specified. Many aspects of a plan, such as the exact rendezvous point of forces, cannot be specified until the plan is underway. Thus, planning and action will occur simultaneously. A particularly important criterion is that potential failures in plans must be detected as early as possible, and plan repairs should be issued before the failure actually occurs.

Planning is a kind of Approximate Processing. Lesser and his colleagues [25] have proposed the idea of approximate processing for real-time systems. The basis of approximate processing is that there are always several methods to achieve any goal, some of which are expensive (in terms of time) but give precise results, and others that are cheaper but less precise. By monitoring and projecting progress, a planner is able to anticipate how long it has before a plan (or plan-fragment) fails or succeeds. It can use this estimate to select among planning methods. An example, illustrated in Section 5 (Figs. 13 and 14), is the granularity of path-planning search: if the planner has relatively little time to find a path between points, it will use a coarse-grained search to get milestones, and then perhaps plan the path in more detail while the bulldozer or crew are in transit to the first milestone.

Planning is Distributed. In organizations, many individuals have responsibility and authority for goals and actions, so planning is distributed and must be coordinated. Sections 6 and 7 illustrate how the envelope mechanism will support distributed planning. Envelopes enable a central node to distribute tasks to autonomous agents, which maintain their own envelopes, and only interrupt the central planner when prespecified boundaries within their envelopes have been crossed (e.g., when an agent's plan is in danger of failing). Envelopes are very similar to the information passed among distributed planners in Durfee and Lesser's partial global planning framework [14].

3 The Relevance of AI Planning and Problem-solving Techniques

Most research on AI planning is only remotely relevant to the operational planning problem. This section discusses the reasons for this and suggests new techniques that are more appropriate²².

3.1 Projective Planning

Early research on planning adopted predicate calculus as a representation language for planners and viewed planning metaphorically (and sometimes literally) as theorem proving. The initial state of the world was captured in a set of axioms, the goal state was a theorem, and the plan was a proof that the goal state could be reached from the initial state. This approach to planning is found in HACKER [34], GPS [29]), STRIPS [15], INTERPLAN [37], WARPLAN [40], Waldinger's planner [39], ABSTRIPS [32], NOAH [33] and NONLIN [36].

From our perspective, the most salient characteristic of these planners was their almost complete avoidance of uncertainty. These programs were crafted under the assumptions that they would have complete, accurate knowledge about the state of the environment, and complete, accurate knowledge about the *immediate* outcomes of all actions. Actions were represented (and indexed) in terms of their immediate outcomes; for example, in the blocks world, one of the immediate outcomes of the action (*take-off x y*) is (*clear-top y*) — removing *x* from *y* clears off the top of *y*. But although these planners knew the immediate outcomes of actions, they were required to search combinatorial spaces of extended ramifications. For example, an action such as (*put-on red-block green-block*) may achieve an immediate goal, but may later impede progress toward a goal that requires *green-block* to have a clear top. A better plan may involve moving *green-block* first and then putting *red-block* on it. Early planners dealt with uncertainty about the extended outcomes of actions by various forms of search. Some algorithms were more efficient than others (i.e., required less backtracking). But all assumed that, because the immediate outcomes of actions are certain, the extended ramifications could be discovered by projection, that is, by internally simulating the actions in a plan before committing to any external actions.

Because planners were assumed to know the current state of the environment and the outcomes of all actions, and because it was assumed that the environment did not change except through the actions of the planner, there was really no need to distinguish internal and external actions. Actions cannot introduce discrepancies between the projected representation of the world and the actual world.

²²This section is adapted from "The centrality of autonomous agents in theories of action under uncertainty," by Paul Cohen and David Day [6]

Clearly, these approaches have little or nothing to do with real-world planning. Operational planning, in particular, is characterized by considerable uncertainty about the effects of actions. This has several sources, as we noted in Section 2.

More recently, AI has developed planning methods that do not depend so heavily on these assumptions. One approach, which modifies the earlier planning algorithms relatively little, involves *replanning* when the environment turns out to be different than projected. For example, Wilkins' SIPE planner was very much like NOAH [sacerdoti75?], but when discrepancies were detected between the environment and its internal representation, SIPE would efficiently modify its plan, maintaining as much of its original plan as possible [42]. Related replanning methods have been developed by [4].

Earlier planners assumed that actions were instantaneous, and that their effects persisted until they were explicitly negated. However, actions take time, and the states they bring about may be ephemeral. Temporal logics and temporal planners address these issues (e.g., see [27,2,12] for work on temporal logic, and [38,19] for temporal planners). Clearly, these techniques are important for operational planning. Planners must always be cognizant of the amount of time it takes to transport troops from one place to another, for instance, or how long a unit can stay fresh and alert.

Just as actions take time in real physical environments, planning and replanning themselves take time. In real-time planning, the agent must allocate a limited resource—time—to planning, replanning, monitoring the environment, and action. Time usually has costs (e.g., in the fire-fighting domain discussed in Sec. 4, forests are consumed while the agent plans.) The balance between internal and external actions is further complicated by uncertainty about the environment: to meet time constraints, agents may begin sequences of actions before they have all the evidence they need; but if they wait, an opportunity may be lost and the evidence will be of no value. Recent work on real-time planning includes that by Durfee [14], Lesser, Durfee and Pavlin [25], Hayes-Roth and her colleagues [20,18,31], Korf [24], Dacus [10], Luhrs, et al. [26], Herman, et al. [23], Daily, et al. [11], Firby and Hanks [17], Hendler and Sanborne [22] and others.

3.2 Reactive Planning

A radically different approach challenges the distinction between planning and execution, and thus the distinction between internal and external action. This view holds that, by any metric, projection in uncertain environments is inefficient: dynamic environments will never be as they are projected to be, so projection is a waste of resources. Projection involves selecting actions that are expected to be appropriate at some point in the future; planning without projection involves selecting actions based on the current, immediate environment, without explicitly considering their consequences. *Reactive planners* do not project, but simply react to their environments, so the distinction between planning and execution is absent [5,1,3,16].

Reactive planning raises questions about the status of goals: planners may *appear* to be goal-directed when, in fact, they are simply responding to their environments. One does not need internal structures called goals to explain apparently intentional behavior. But we believe that intelligent agents should reason about their goals, so some goal-directed behavior will not be generated by reactive planning [28,13].

We believe reactive planning is an extreme response to uncertainty in the environment. We agree that the value of projection depends on the certainty with which we can predict the outcomes of actions; but since this is variable, and depends on many factors, we do not agree that planners should completely forego projection. (Others have made similar observations [21,13,35].)

We will assess the relevance of reactive planning to operational planning in Section 8, after we have introduced the notion of envelopes (Sec. 6). Suffice it to say that the command structure of military organizations seems to be set up so that higher levels can view lower levels as reactive. Orders seem to specify, at some fairly gross level of abstraction, how lower levels should respond to changes in their environment. This makes them reactive and also, to some degree, predictable. We will return to this point in Section 8.

3.3 Probabilistic Planning

We can, in fact, project even if we do not know the precise outcomes of actions. Decision analysis is a form of planning by projection: When actions have uncertain outcomes, and these lead to further actions, then a combinatorial space of actions and outcomes is quickly generated. If one knows the probabilities that actions will lead to particular outcomes, and also the utilities of the outcomes, then one can find the *subjective expected utility* of actions. For example, imagine test-1 costs \$10 and will accurately say whether or not a patient has disease A, but says nothing about diseases B and C; and test-2 costs \$50 and is diagnostic for disease B but says nothing about A or C. Which diagnostic action should we take first, test-1 or test-2? Assuming A, B, and C have equal priors, it is most efficient to do test-1 first. Decision analysis will resolve the question for the general case of unequal priors.²³

But note that two assumptions are implicit in the example: the statement of the problem implies that the hypotheses A, B, and C are both *mutually exclusive* and *exhaustive*. Thus, if test-1 finds A, we need not do test-2; and if both tests fail to find A and B, then the answer must be C. Mutual exclusivity is a special case of conditional probability: When we say A and B are mutually exclusive we mean that the probability of disease B is conditional on the outcome of test-1 (and equivalently, our belief in A), and vice-versa. Thus, in the general case, to plan a sequence of tests to find out which disease a patient has, we need to know the conditional probability of each disease given each combination of outcomes of tests for these diseases. Even if we assume the diseases are mutually exclusive and exhaustive, and we assume that every test

²³We are grateful to Professor Glenn Shafer for this example.

either confirms or disconfirms a disease (but does not provide partial support for any disease), we are still faced with a combinatorial search because there are $N!$ sequences of diagnostic actions, and because each action can have several possible outcomes.

We are not rejecting decision analysis as a technique for planning under uncertainty, only noting that its inherent combinatorics must be managed carefully (e.g., Wellman has proposed some approximate forms of decision analysis [41]). All planning by projection generates combinatorial spaces of plans; uncertainty about outcomes simply makes the problem worse. We expect that decision analysis can be merged with AI planning techniques (e.g., least commitment and hierarchical planning) to reduce the combinatorics of projection. For example, hierarchical planning reduces the combinatorics of projection by generating plans at successive levels of abstraction [7]. Decision analysis might be used to select actions at each level.

In Section 8, we discuss the power of abstraction in managing the combinatorics of projection in operational planning. The same point holds for decision analysis, which is just a probabilistic projection: If the actions of military organizations are viewed at a sufficiently gross level, one can in fact project the progress of engagements to some point in the future.

3.4 Local Search

Operational planning is in many respects a resource allocation problem. This is true for both the static and dynamic aspects of the problem. In the static case, we need to assign resources to geographic areas to contain an enemy force; in the dynamic case, we must reason carefully about our materiel and other physical resources, and also our temporal and computational resources. We are thus very interested in a class of operations research techniques called *local search* methods. The idea of local search is simple: One first generates an approximate solution to a resource allocation and then, by a series of "swaps" transforms the solution iteratively into a better one. Expert heuristics can be integrated with local search as follows: swaps are dictated not by a domain-independent evaluation metric, but by expert knowledge about how to transform one solution into a better one. This technique seems especially well-suited to the static phase of operational planning, in which military forces are allocated to geographic areas in accordance with constraints. We imagine that an algorithm would generate the first solution, the first allocation of forces, and then expert heuristics would be used to permute this into a better solution. To date, these techniques have been used for layout problems (e.g., laying out floor plans subject to numerous, sometimes conflicting constraints). We believe they will also be useful for the spatial "layout" of forces.

4 The Fire Problem.

For the last eighteen months, one of us has been directing the development of the PHOENIX planner for controlling forest fires. We have built a large simulation of forest fires and the

equipment commonly used to put them out. PHOENIX, which is still incomplete, operates in this dynamic, real-time world. Its goal is to manage the fire—limit the loss of human life, limit the damage to forest and buildings, and limit the monetary costs of achieving these goals. This task closely approximates the problems faced by a forest fire manager, but more importantly, it is representative of a class of military command and control problems.

Our simulation consists of a large geographical area ("Explorer National Park") in which there is a considerable variety of topography and ground cover, as well as roads, lakes, and streams. Very recently, we have obtained data from the Defense Mapping Agency, so PHOENIX is now able to plan using all the terrain features that military planners consider important. These features affect how forest fires burn. Equally important features are wind speed and direction, both of which can change unpredictably, and the moisture content of the ground cover, which varies in time and geographically. To fight the fire, the simulation provides bulldozers, crews, transport vehicles, planes and helicopters. These cut fire line, move firefighters, spray water, or dump retardant.

These fire-fighting agents can be directed either by an automatic planner or else by a human player of what is essentially a complex, real-time video game. We have already gained considerable insight into (and respect for) the dynamics of this mini-world by playing against the simulation—often losing many lives and considerable real estate to a seemingly slow and containable fire. It is difficult for a planner, human or AI program, to do very well at the game (i.e., put out the fire with reasonable costs, no loss of life, etc.) because:

- The player's knowledge of the fire is limited to what the agents in the field can "see." Crews and bulldozers can see only short distances; aircraft can see further. The planner rarely, if ever, has complete knowledge of the extent or location of the fire.
- The behavior of the fire cannot be accurately predicted because some factors that affect it, terrain, ground cover and the moisture content of the ground cover, are known only approximately. Moreover, wind speed and direction can change unpredictably.
- The behavior of the fire-fighting agents cannot be accurately predicted. In particular, the time required to move to a location or perform some task depends on terrain and ground cover. Fire-fighting agents also have limited autonomy to run away from a fire, so the central planner cannot always be sure of their location.
- The simulation is real-time with respect to the fire. While fire-fighting agents move, cut line and drop retardant, the fire keeps burning. Most important, any time the planner devotes to deliberation is claimed as real estate by the fire.

The environment with which the planner interacts is independent, dynamic and probabilistic. It is independent because the planner is not the only agent of change, dynamic because changes take place over time, and probabilistic because the magnitude and temporal extent of changes to the environment are unpredictable.

To plan in this environment an agent needs to know when to project and when to plan reactively. Projection is desirable to avoid pitfalls, and to increase the efficiency of plans. But projection itself takes time, and uncertainty precludes avoiding *all* pitfalls, and efficient plans are useless if no time remains to execute them; so there will be situations when the timeliness of reaction will make the difference between success and failure.

We now describe the architecture of the PHOENIX planner, which is under construction for this task. We defer the discussion of the parallels between this task and military planning until Section 8.

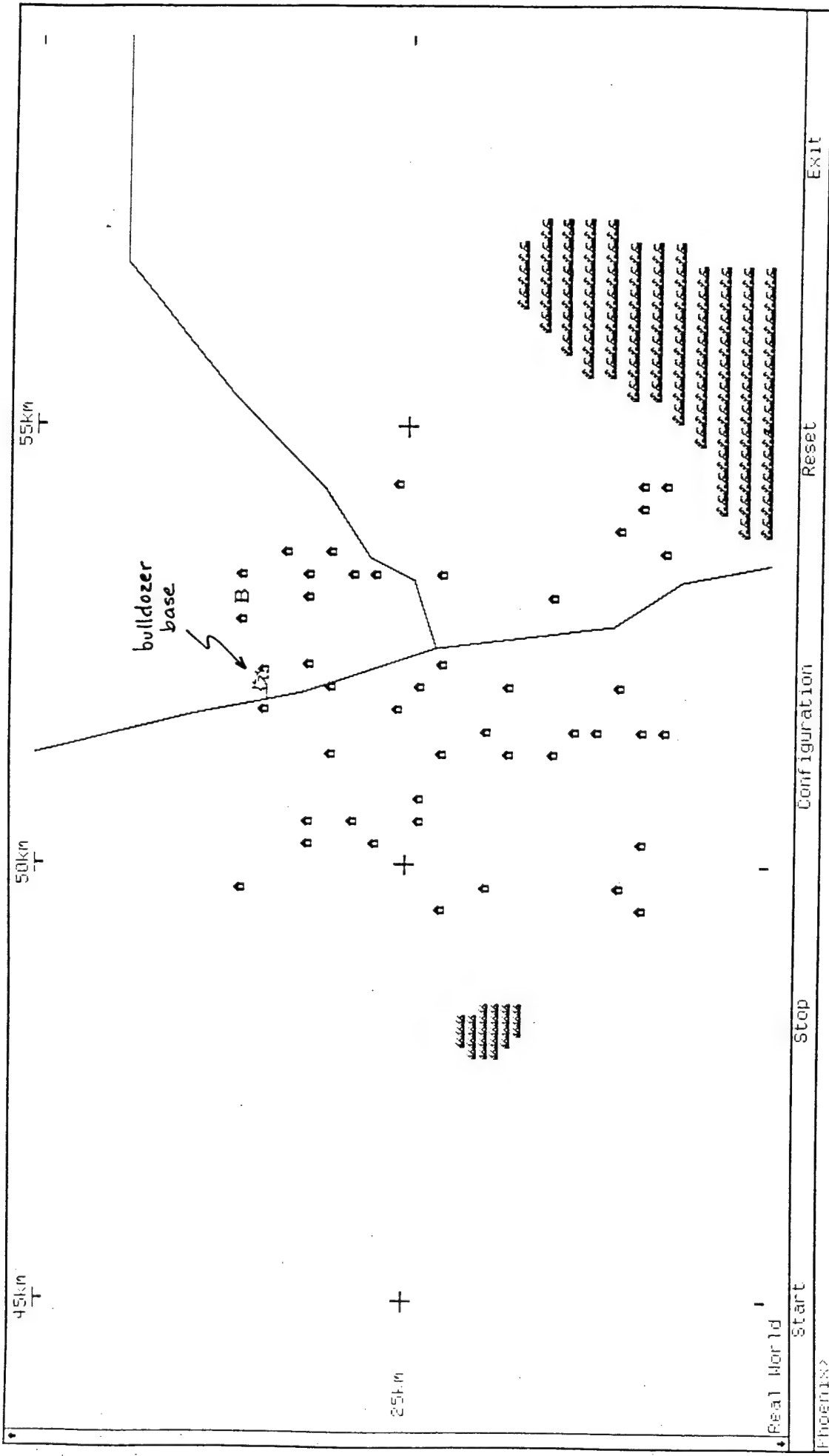
5 The PHOENIX Architecture

To our knowledge, the literature on operational planning, including doctrine and training texts, contains very little on the problems and processes of plan monitoring and plan revision. Clearly, these are extremely important aspects of the operational planning task. They are addressed in courses at the Command and General Staff College, but since there are no formal models of plan monitoring and revision, both pedagogy and knowledge engineering are impeded. This section presents one such model—an AI planner for real-time control of forest fires. It generates plans and monitors them during execution. The model is designed to permit plan revision, though this aspect has not been implemented yet.

The PHOENIX architecture is designed for real-time, incremental planning with approximate processing. This means that it begins executing plans before planning is finished, and that it selects from among problem solving methods those that give the best results in the allowed time. In the following subsections, we illustrate PHOENIX in operation, giving enough detail to see roughly how PHOENIX works, and then discuss its architecture.

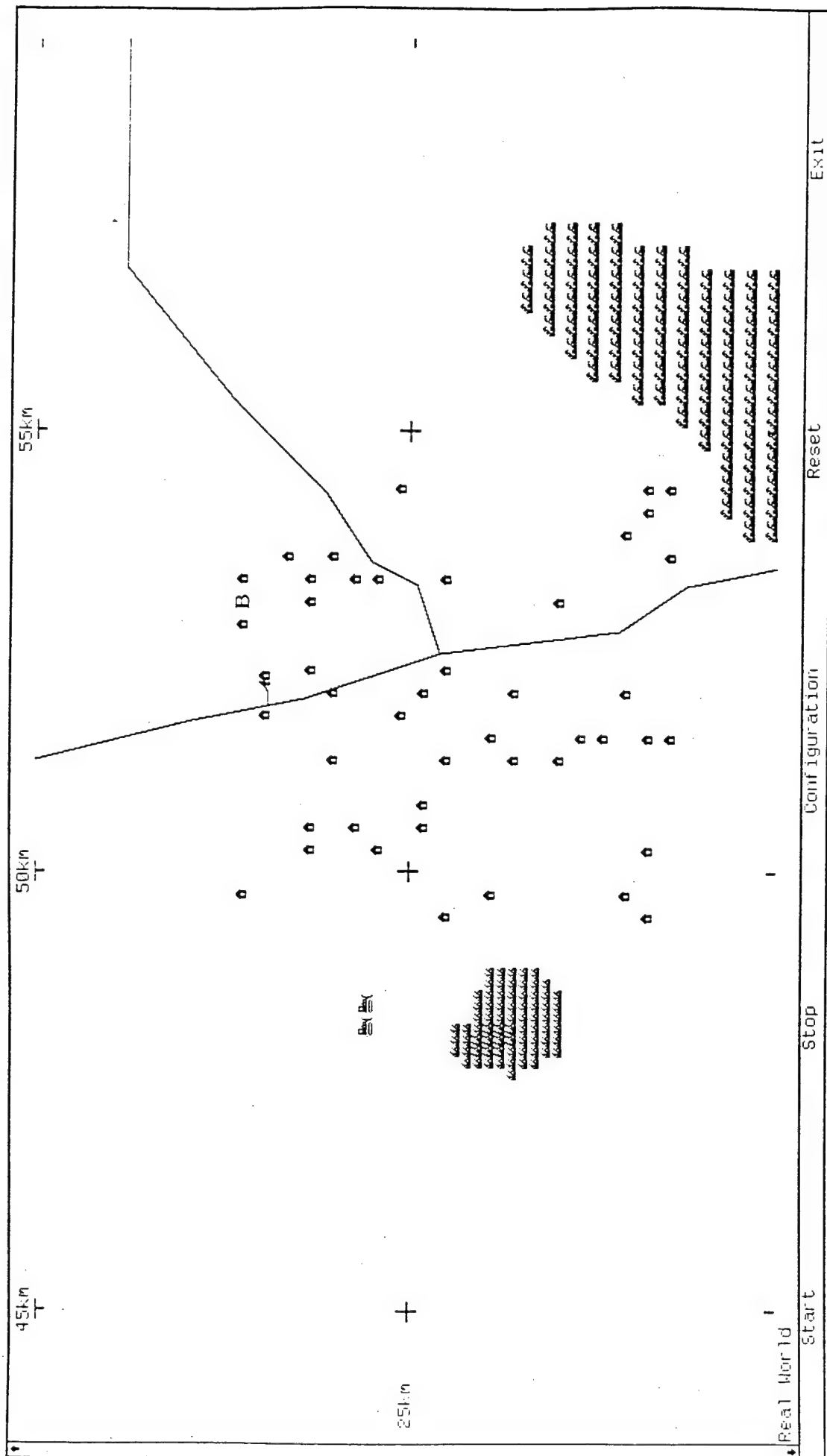
The following pages illustrate aspects of the PHOENIX. All of the illustrations are taken directly from the screen of the TI Explorer. It should be noted that these illustrations are very sparse compared the system we have running on color monitors. We use color to convey groundcover and elevation information in the firemap, as well as highlighting and temporal information in the desktop (knowledge engineering) mode. Unfortunately, this information is lost in the transfer to a black-and-white printer.

Figure 1. The forest-fire simulator with a fire in progress (center left). The upper pane in this display is the map of the forest. The lower panes allow interactive input and show the status of various system parameters (time of day, current simulation step, etc.). The map is marked with a distance grid (in kilometers), with the top left corner as the origin. The intersections of these grid lines are marked with "+" marks. This part of the map shows a populated area near a lake (lower right) at the intersection of two roads. There are two bulldozers based near one of the roads (upper center), as well as the fireboss control center, marked by a "B". At this time, the fire has been spotted and the fireboss is sending the bulldozers to the fire (notice the status message in bottom right pane).



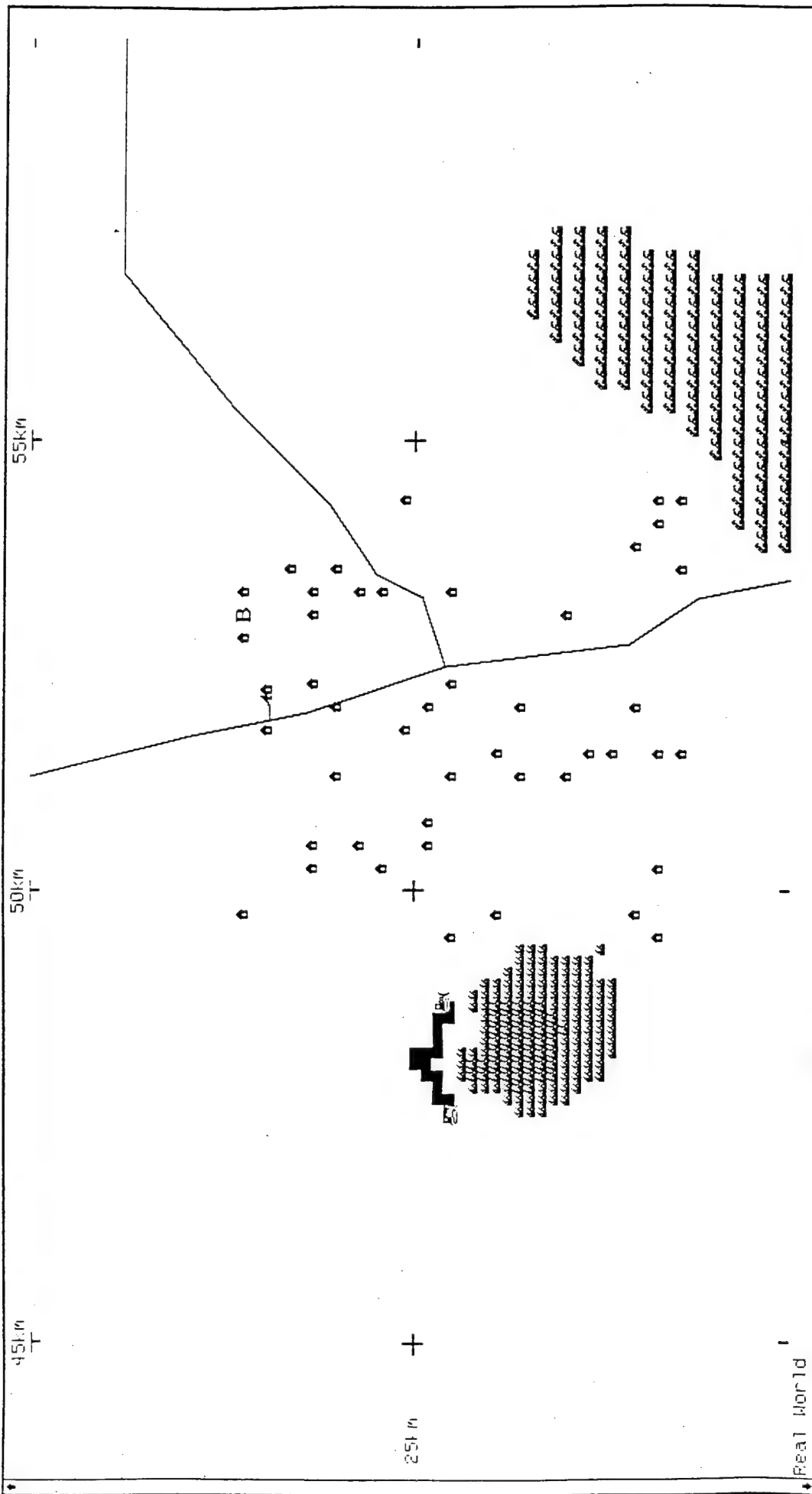
- Figure 1-

Figure 2. Two bulldozers arriving at the fire. Notice the spread of the fire and the time that has elapsed since Figure 1 (simulation time in bottom left pane).



- Figure 2 -

Figure 3. The bulldozers have begun to encircle the fire with fireline. They started digging at the same point and are surrounding the fire in opposite directions. Notice that the center of the fire is now burning hotter than the outside.

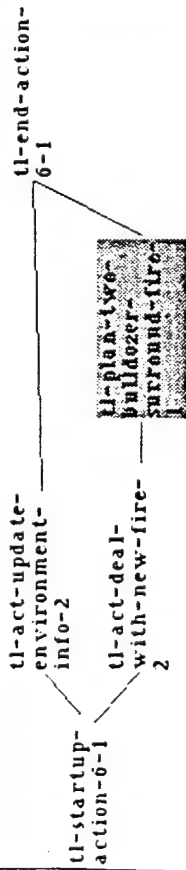


Phoenix

- Figure 3 -

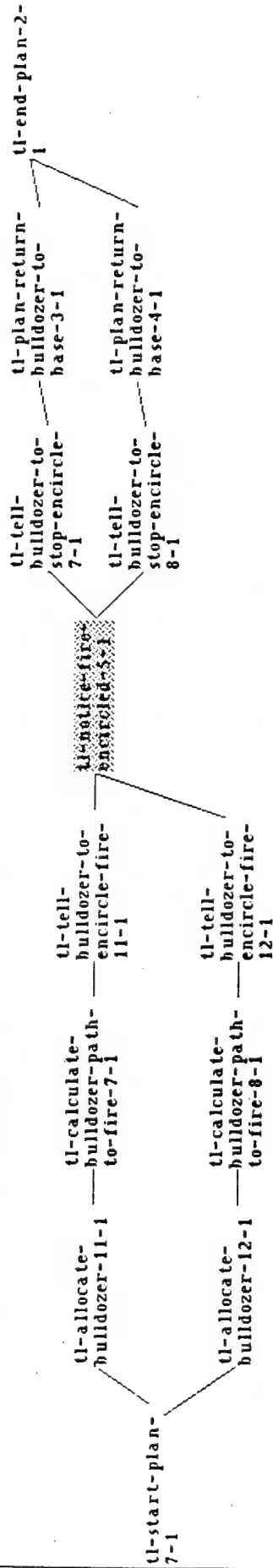
Figure 4. This figure illustrates the Desktop mode of the planning system, a knowledge engineering interface in which the various knowledge representations used in the system can be graphed, edited, and debugged. These graphs show the cognitive timeline of the fireboss, where the current plan for fighting the fire is represented as a temporal partial ordering of plan steps. The graph in Fig. 4a is the top level of the plan. At this level the fireboss is following two parallel courses of action: the environment is being monitored (for new fires and for changes that may affect the current plan), and the fire that has been reported is being fought. This second course of action is done in two successive steps: a plan is chosen and put on the timeline, and then is executed. The graph in Fig. 4b is an expansion of step that is highlighted in Fig. 4a. This is the active plan step in the top level plan, and is itself a plan called two-bulldozer-surround-fire. The current step in this lower level plan is highlighted; all preceding steps (to the left) have been executed. Those steps allocated the two bulldozers and sent them to the fire. The current action is waiting to notice that the fire has been encircled. The subsequent steps will tell the bulldozers to stop digging line and return to the base.

Timeline: initial-timeline-6 in fireboss-2



(a)

Plan: tl-plan-two-bulldozer-surround-fire-1 in fireboss-2



(b)

Exit

Reset

Configuration

Stop

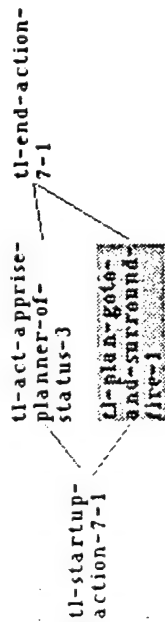
Start

Phoenix>

- Figure 4 -

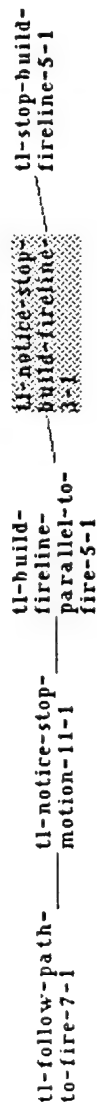
Figure 5. These graphs show the cognitive timeline of one of the bulldozers (viewed at the same simulation time as the fireboss's timeline graphs). Each bulldozer has the same architecture as the fireboss, with its own planning knowledge and timeline. The differences between the plan knowledge of the fireboss and the bulldozers is contextual; the bulldozers only know about plans governing their own activities, whereas the fireboss has the more sophisticated plan knowledge needed to coordinate multiple agents. This architectural consistency will provide the ability to do research in distributed planning among cooperating agents, since all agents have the same cognitive structure (we currently use a centralized model where the fireboss coordinates agents). The top level of the current bulldozer plan in Fig. 5a includes the parallel actions of surrounding the fire and sending periodic status updates to the fireboss. The highlighted action expands into the lower level plan steps for surrounding the fire, as we see in Fig. 5b. The current action in this plan is to receive a message from the fireboss to stop building fireline. This message will be sent by the next action(s) in the fireboss's cognitive timeline (see Fig. 4b), the step tell-bulldozer-to-stop-encircling.

Timeline: initial-timeline-7 in bulldozer-3



(a)

Plan: tl-plan-goto-and-surround-fire-1 in bulldozer-3



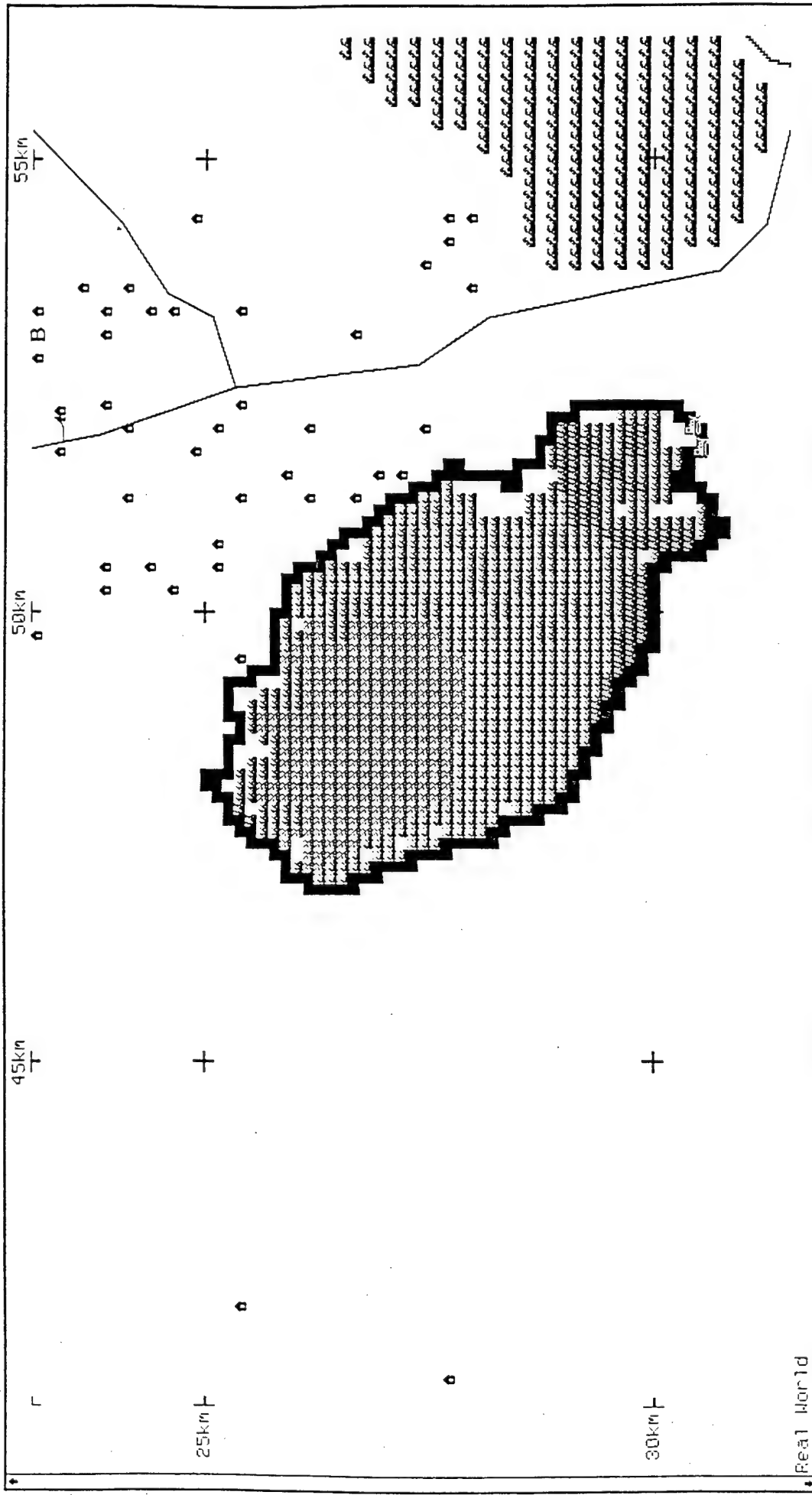
(b)

Start Stop Configuration Reset Exit

Phoenix> desktop
Phoenix> Set-Frame-System
Phoenix> Display-Timeline
Phoenix> Set-Frame-System
Phoenix> Display-Timeline
Phoenix> Display-Timeline
Phoenix>

- Figure 5 -

Figure 6. The bulldozers have encircled the fire. Notice the extent of the fire spread and the amount of simulation time that has elapsed.



Exit

Reset

Configuration

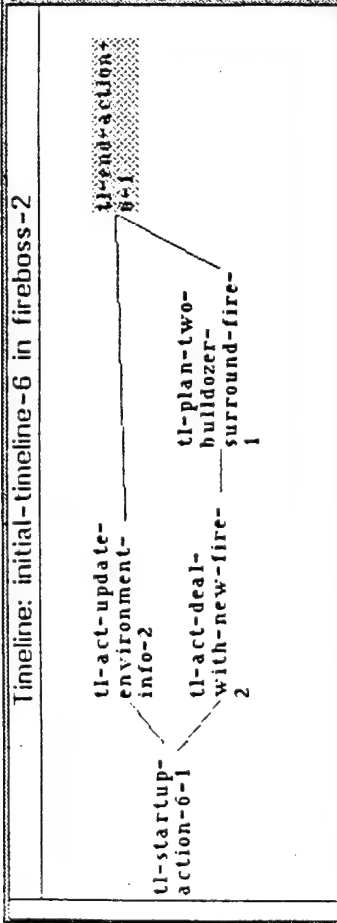
Stop

Start

Phoenix>

- Figure 6 -

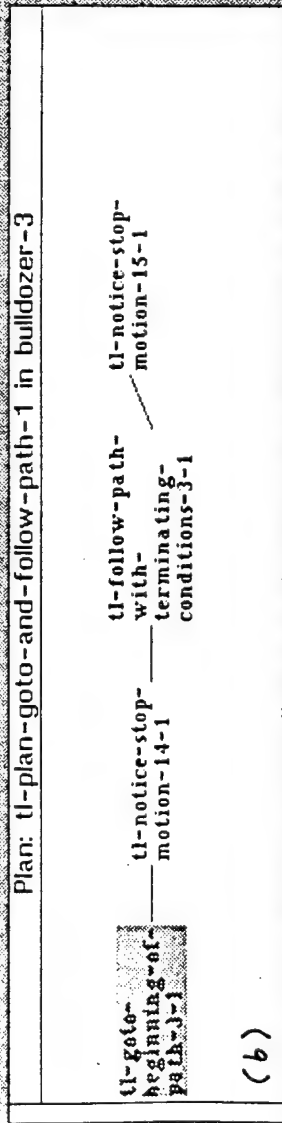
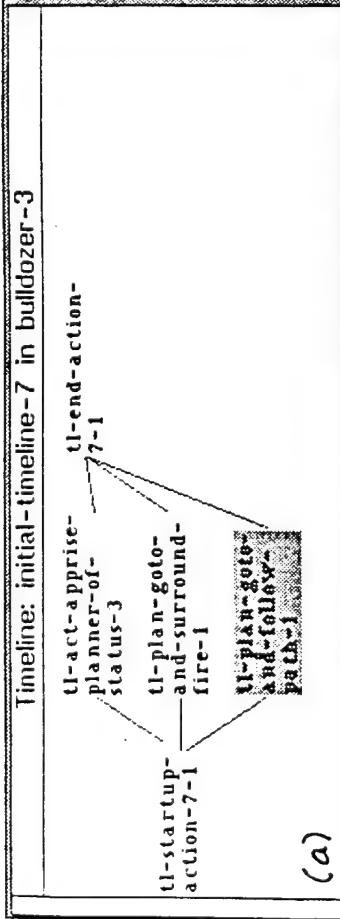
Figure 7. At this point, the fireboss has instructed the bulldozers to return to base (see the penultimate step in Fig. 4b). This completes all active plans, so its timeline is at the highlighted termination action.



Start	Stop	Configuration	Reset	Exit
Phoenix> Desktop				
Phoenix> Set-Frame-System				
Phoenix> Display-Timeline				
Phoenix>				

- Figure 7 -

Figure 8. The bulldozer is still busy executing its plan to return to base, as illustrated by these timeline graphs. The plan step to return to base has been added to the top level timeline. (Though it's not clear, the actions to apprise-planner-of-status and surround-the-fire are no longer active - they've been completed. This information is color-coded in the color version of the system). The graph in Fig. 8b expands this action with a plan for returning to base. It is currently executing the highlighted step to go to the beginning of its return path. Figure 9. The remaining figures show frames for various cognitive, state, and data structures in the system. They are presented in Desktop mode in the frame editor, where their attributes and relations can be viewed and modified. It should be noted that all objects in the system are represented as frames for modularity and portability.



Start	Stop	Configuration	Reset	Exit
Phoenix> Desktop				
Phoenix> Set-Frame-System				
Phoenix> Display-Timeline				
Phoenix> Set-Frame-System				
Phoenix> Display-Timeline				
Phoenix>				

- Figure 8 -

Fig. 9 shows the frame representing the plan that was executed in the previous figures. This frame has information about the type of plan (A-Kind-Of slot), the key for matching a plan to the context in which it is applicable, the variables used by the plan, and the steps in the plan. There is also information about how to construct the cognitive timeline for the plan. This plan resides in the fireboss's Plan Library, which is a collection of frames representing the plans known to it. When a new fire is sighted, the fireboss assesses the conditions (e.g., wind speed, direction, terrain type, available resources, etc.) and selects a skeletal plan from its Plan Library. It then instantiates this plan for the particular situation, adds it to the timeline, and begins to execute it. This much of the planner has been implemented. Monitoring with envelopes and plan modification require further research.

Figure 10. These frames represent the cognitive timeline of the fireboss (see Fig. 4). Initial-timeline-6 is the top level plan (Fig. 10a), and includes the information needed to set up the top level timeline. Tl-plan-two-bulldozer-surround-fire-1 (Fig. 10b) represents the expansion of the plan for two bulldozers to surround the fire. Notice the slots for temporal values and informational messages from the bulldozers.

Initial-timeline-6 in fireboss-2

Mode: Display all slots

Has-candidate-action

Values

Tl-end-action-6-1

Tl-act-update-environment-info-2

Has-component

Values

Tl-plan-two-bulldozer-surround-fire-1

Tl-act-deal-with-new-fire-2

Tl-act-update-environment-info-2

Tl-startup-action-6-1

Tl-end-action-6-1

Has-end-action

Values

Tl-end-action-6-1

Has-start-action

Values

Tl-startup-action-6-1

Instance

Values

Initial-timeline

(a)

Tl-plan-two-bulldozer-surround-fire-1 in fireboss-2

Mode: Display all slots

Candidate-action-of

Values

Component-of

Values

Initial-timeline-6

Creation-time

Values

3900

Executing-action-of

Values

Execution-time

Values

3900

Has-component

Values

Tl-start-plan-7-1

Tl-allocate-bulldozer-11-1

Tl-calculate-bulldozer-path-to-fire-7-1

Tl-tell-bulldozer-to-encircle-fire-11-1

Tl-allocate-bulldozer-12-1

Tl-calculate-bulldozer-path-to-fire-8-1

Tl-tell-bulldozer-to-encircle-fire-12-1

Tl-notice-fire-encircled-5-1

Tl-tell-bulldozer-to-stop-encircle-7-1

Tl-plan-return-bulldozer-to-base-3-1

Tl-tell-bulldozer-to-stop-encircle-8-1

Tl-plan-return-bulldozer-to-base-4-1

Tl-end-plan-2-1

Has-data-frame

Values

Data-frame-2

Has-end-action

Values

Tl-end-plan-2-1

Has-information-message

Values

Information-message-4

Information-message-3

Has-start-action

Values

Tl-start-plan-7-1

Instance

Values

Timeline-entry

Plan-two-bulldozer-surround-fire

(b)

Start

Stop

Configuration




Reset

Exit

Phoenix>

- Figure 10-

Figure 11. These frames represent the relationship between a step in a timeline (as in the previous figure) and the methods used to execute that step. The frame Tl-calculate-bulldozer-path-to-fire-7 (Fig. 11a) represents the step by the same name from the timeline in Fig. 10b. This frame is used to place the action step of calculating a path on a timeline, and is A-Kind-Of the action Act-calculate- bulldozer-path-to-fire. This action is a first order object in our evolving planning language, and as such is represented by its own frame (Fig. 11b). This frame contains information about this generic action, such as the variables used and the timeline steps currently using this action frame. It also includes a slot for the execution method(s) associated with the action. This is one method we will use to experiment with real-time control of agents' cognitive activities, as we allow each action a range of execution methods (enumerated in this slot) which vary in their contextual applicability according to such features as time-required, quality-of- solution, etc. An execution method is represented as another frame (Fig. 11c), which has a pointer to a specific method for performing a given action. It will eventually hold the criteria used by the filtering mechanism that chooses which execution method for an action is most appropriate under the current circumstances.

	<div data-bbox="459 1081 966 2001"> <div>  Tl-calculate-bulldozer-path-to-fire-7 in fireboss-2 </div> <div> Mode: Display all slots </div> <div> <div>Alko</div> <div>Values Timeline-entry Act-calculate-bulldozer-path-to-fire</div> <div>Component-of</div> <div>Values Plan-two-bulldozer-surround-fire</div> <div>Initial-frame</div> <div>Values T</div> <div>Instance+inv</div> <div>Values Tl-calculate-bulldozer-path-to-fire-7-1</div> <div>Next-action</div> <div>Values Tl-tell-bulldozer-to-encircle-fire-11</div> <div>Previous-action</div> <div>Values Tl-allocate-bulldozer-11</div> <div>Variable-mapping</div> <div>Values (Bulldozer bulldozer1 path path1)</div> </div> <div>(a)</div> </div> <div data-bbox="167 184 654 989"> <div>  Act-calculate-bulldozer-path-to-fire in fireboss-2 </div> <div> Mode: Display all slots </div> <div> <div> <div>Action-variables</div> <div>Values Plan-id Bulldozer Fire Path</div> </div> <div>Alko</div> <div>Values Action</div> <div>Alko+inv</div> <div>Values</div> <div> <div>Tl-calculate-bulldozer-path-to-fire-10</div> <div>Tl-calculate-bulldozer-path-to-fire-9</div> <div>Tl-calculate-bulldozer-path-to-fire-8</div> <div>Tl-calculate-bulldozer-path-to-fire-7</div> <div>Tl-calculate-bulldozer-path-to-fire-6</div> </div> <div>Has-execution-method</div> <div>Values Em-calculate-bulldozer-path-to-fire</div> <div>Initial-frame</div> <div>Values T</div> </div> <div>(b)</div> </div> <div data-bbox="764 184 1109 989"> <div>  Em-calculate-bulldozer-path-to-fire in fireboss-2 </div> <div> Mode: Display all slots </div> <div> <div>Alko</div> <div>Values Execution-method</div> <div>Execution-method-of</div> <div>Values Act-calculate-bulldozer-path-to-fire</div> <div>Initial-frame</div> <div>Values T</div> <div>Method</div> <div>Values :Em-calculate-bulldozer-path-to-fire</div> </div> <div>(c)</div> </div>	<div data-bbox="1222 84 1542 2020"> <div>Phoenix></div> <div> <div>Start</div> <div>Stop</div> <div>Configuration</div> <div>Reset</div> <div>Exit</div> </div> </div> <div data-bbox="1498 84 1542 2020"> <div>8/2 3:30 fireboss-2 Error recovery 8/2 3:30 An error occurred in bulldozer-3></div> <div>32A</div> </div>
--	--	---

- Figure 11 -

Figure 12. This figure illustrates the use of frames to represent all system objects. The frame in Fig. 12a represents what the fireboss knows about the fire. This frame is part of the fireboss's state memory, and holds only the fragmentary knowledge gained through reported observations by agents such as watchtowers and bulldozers (as opposed to the complete knowledge about the fire's characteristics available to the simulator). The frame in Fig. 12b illustrates a communication channel between agents in the system. It shows an information message from a bulldozer to the fireboss, including the message, a timestamp, and the method of communication (walkie-talkie in this case). Message handling is an important function in the fire-fighting system. Everything the fireboss knows comes from communications with other agents; it bases its choice of plans and coordination of agents on the resulting picture it constructs of the fire. When we begin to look at distributed control in the system, communication between autonomous agents will become multi-dimensional, as any agent might potentially communicate with any other (not just the fireboss) in order to coordinate sensory data fusion and cooperative courses of action. The frames in Fig. 12c and 12d represent the fireboss and a bulldozer. They are used to store knowledge about these agents and pointers to their state memories and cognitive structures (plan libraries, timelines, sensory/motor capacities).

Fire-2 in fireboss-2	
Mode: Display all slots	
Center-of-mass	
Values (49429 . 27796)	
Height	
Values 3635	
Instance	
Values Fire	
Height	
Values 872	
Orientation	
Values 0.7853982	
Origin	
Values (43650 . 24438)	
Sums	
Values (43102238 . 24238165)	
Width	
Values 6403	

(a)

Fireboss-2 in fireboss-2	
Mode: Display all slots	
Environment-info	
Values Environment-info-2	
Instance	
Values Fireboss	
Plan-id->plan-instance-mapping	
Values	
(fireboss-2-plan-id-3	
tl-plan-return-bulldozer-to-base-3-1	
fireboss-2-plan-id-2	
tl-plan-return-bulldozer-to-base-4-1	
fireboss-2-plan-id-1	
tl-plan-two-bulldozer-surround-fire-1)	

(c)

Information-message-3 in fireboss-2	
Mode: Display all slots	
Communication-channel	
Values Mt	
From	
Values Bulldozer-3	
Information-message-of	
Values Tl-plan-two-bulldozer-surround-fire-1	
Instance	
Values Information-message	
Plan-id	
Values Fireboss-2-plan-id-1	
Receive-time	
Values 9318	
Send-time	
Values 9318	
Start-build-line-position	
Values (48113 . 25024)	

(b)

Bulldozer-3 in fireboss-2	
Mode: Display all slots	
Display-instance	
Values #Icon: bulldozer-3>	
Initial-position	
Values (52106 . 23372)	
Instance	
Values Bulldozer	
Last-update-time	
Values 55206	
Position	
Values (51942 . 30517)	
Status	
Values Allocated	

(d)

Exit

Reset

Configuration

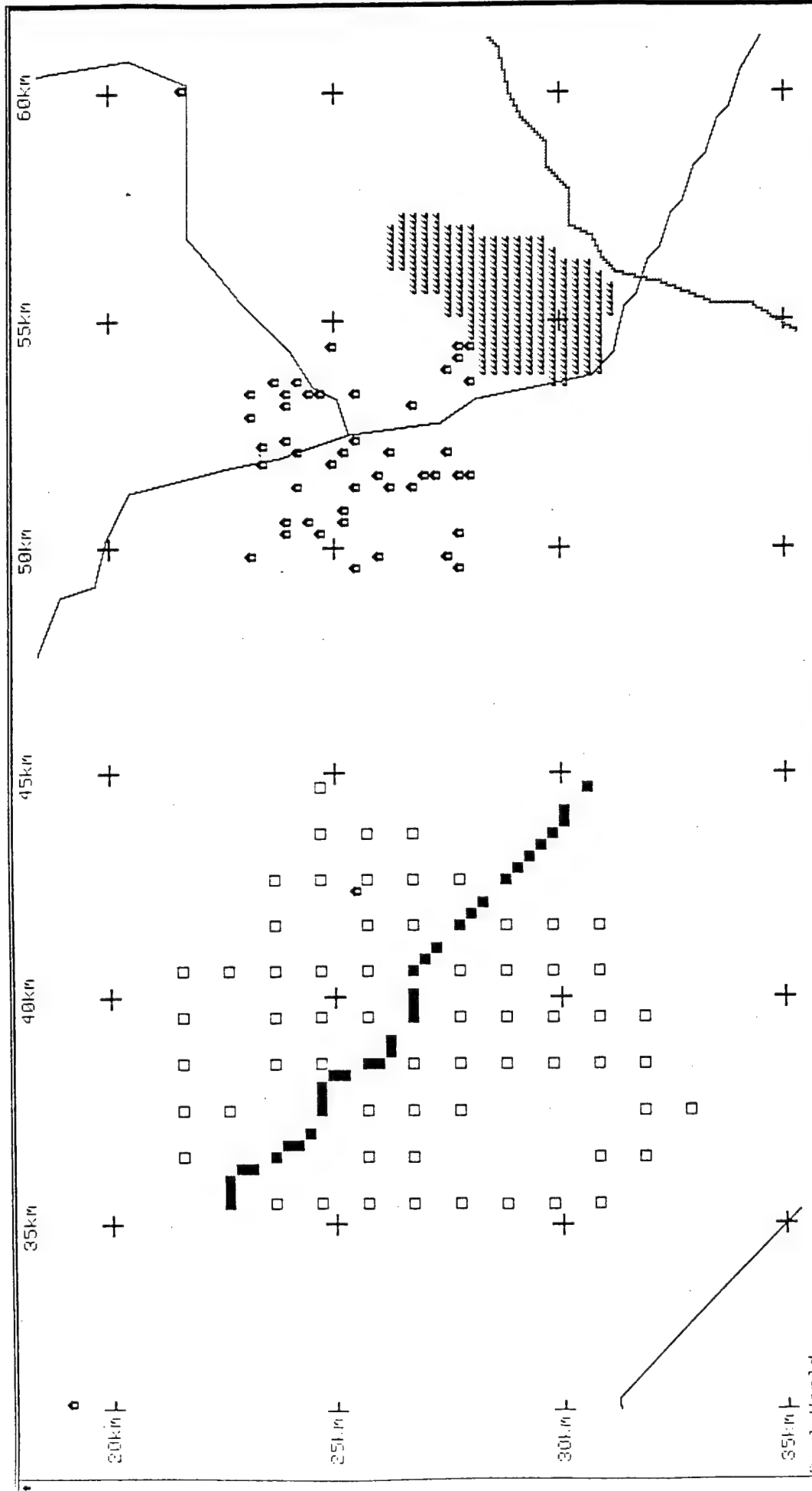
Stop

Start

Phoenix

- Figure 12 -

Figures 13 and 14. These figures show the search conducted by a modified A* path-planning algorithm. In Figure 13, the search is at a coarse level of granularity ($2 + 11$ meters), whereas in Figure 14 the search is an order of magnitude more refined ($2 + 10$ meters.) The search in Figure 13 takes less time, but finds a longer path than the search in Figure 14. This is a simple example of approximate processing. The idea of approximate processing is that every goal can be accomplished in several ways, some expensive and precise, and some cheap but less good. In a dynamic, real-time, unpredictable environment like the fire, one cannot hope to schedule all problem-solving activities in advance, but instead must schedule and reschedule dynamically. Approximate processing provides necessary degrees of freedom; for example if the planner is approaching a deadline, it has the flexibility to choose a coarser search (Fig. 13) than it would ideally like.



Start	Stop	Configuration	Reset	Exit
<ul style="list-style-type: none"> Real World Phoenix > (1) Phoenix > (h1 :abstraction-level 10) Phoenix > (h1 :abstraction-level 10) Phoenix 				
<p>I. Hierarchical*least-commitment A* (searches in the 1/2 plane towards the goal) Abstraction level: 2¹⁰ meters</p> <p>- Figure 14 -</p>				
8/1 12:00	Wait for H-step	8/1 12:00 <Starting scheduler>		35B

5.1 Implementation of PHOENIX

The Phoenix environment has two major components, a discrete event simulation kernel (Fig. 15.c) and an agent architecture (Fig. 15.b). From these, it is possible to build organizations of fire-fighting agents (Fig. 15.a), or, as noted below, any other kinds of agents that operate in map-like environments. The kernel provides the illusion of parallel, continuous activity on a serial machine (Fig. 15.c). It supports the definition of spatial worlds such as the forest, and processes, such as fires, planning, sensors, effectors, and reactions. It provides a low-level task scheduler that automatically keeps all these processes coordinated. The user never sees the kernel, and can think of all processes as simultaneous.

Currently, the Phoenix environment simulates forest fires in the Yellowstone National Park, as discussed in the previous subsection. But the Phoenix discrete event simulation kernel is generic. It can manage any simulations that involve maps and processes. For example, we could replace the forest fire environment with an oil-spill environment. We could replace our map of Yellowstone with oceanographic maps of, say, Prince William Sound. Fire processes have spatial extent, and spread according to wind speed, direction, fuel type, terrain, and so on. They could easily be replaced with oil-slick processes, which also have spatial extent, and spread according to other rules. Similarly, we could replace the definitions of bulldozers and airplanes with definitions of boats and booms.

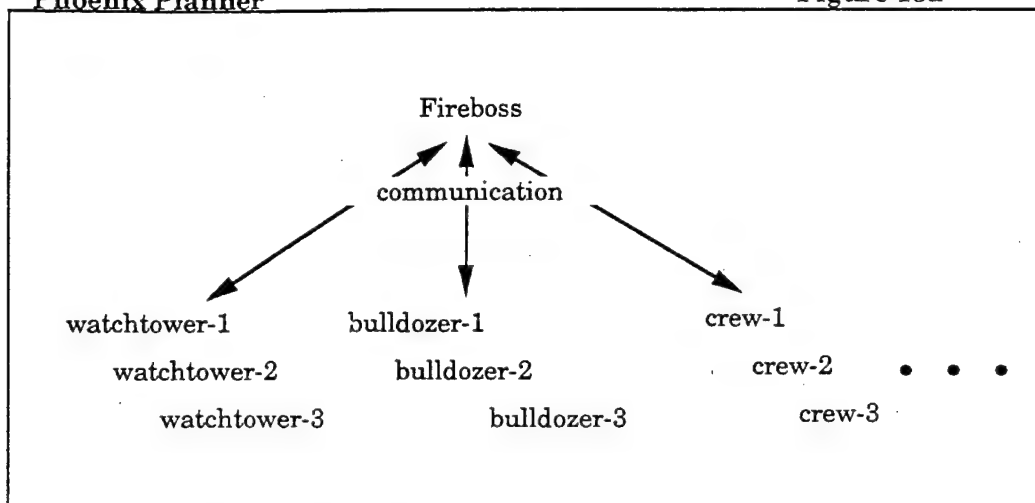
Just as one can change the world by replacing maps and processes in the Phoenix discrete event simulator, so can one change (and thus experiment with) planning methods within the generic Phoenix agent architecture (Fig. 15.b). The agent architecture assumes that agents have sensors and effectors, reactions, and a cognitive component, but it doesn't stipulate how fast they run, how much they know, or how they work. Thus the agent architecture is a framework in which to design and build agents. It doesn't require these components for any given agent, but it does facilitate their definition.

There are four components to the architecture of an agent (Fig. 15.b). Sensors perceive the world. Each agent has a set of sensors for such perceptions as fire- location (are any cells within my radius-of-view on fire?). Effectors perform physical acts such as moving or digging fireline. Reactions are simple stimulus- response actions, triggered when the agent is required to act faster than the time-scale of the cognitive component can handle. An example is the reaction by a bulldozer to stop if it is moving into the fire. The cognitive component performs mental tasks such as planning, monitoring actions, evaluating perceptions, and communicating with other agents. It is composed of a number of subcomponents (see below for details). Each agent has these four components; each component can be endowed with a range of capabilities from limited (or none) to sophisticated.

The cognitive component of an agent performs tasks such as selecting and instantiating plans, monitoring, projection, and reasoning about the use of cognitive resources (notably time). It is composed of five subcomponents. State memory stores known information about the world, such as sensory reports and event histories. It also has pointers to the maps representing static

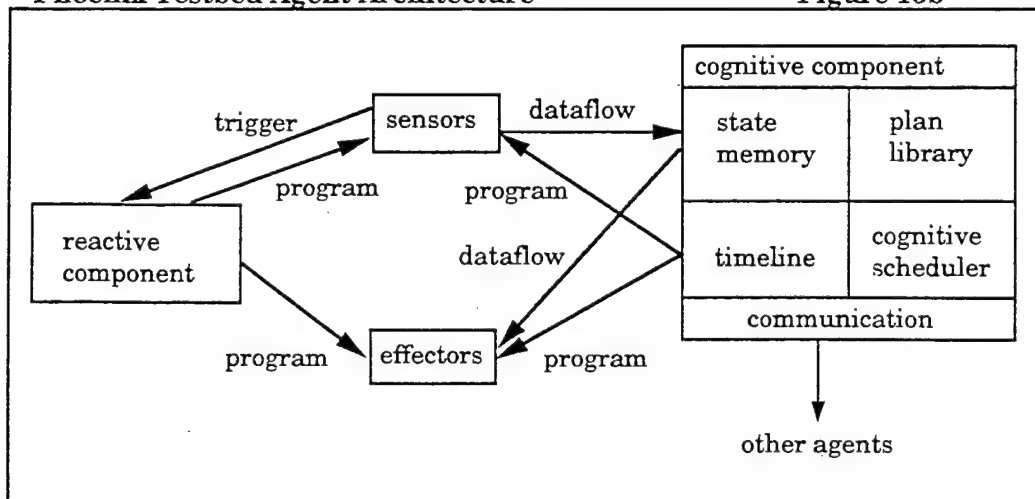
Phoenix Planner

Figure 15a



Phoenix Testbed Agent Architecture

Figure 15b



Phoenix Testbed Discrete Event Simulator

Figure 15c

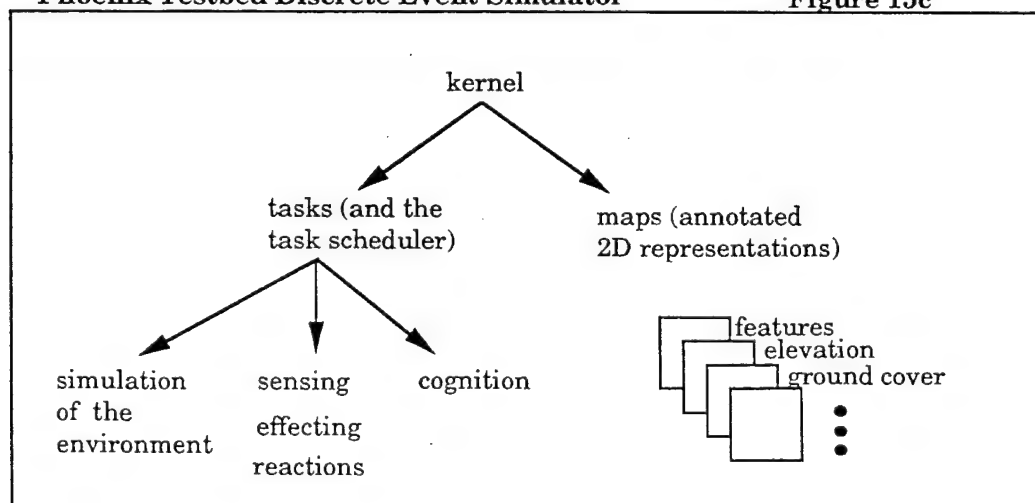


Figure 15. Three levels of the Phoenix Architecture

characteristics of the simulation world. The plan library holds plan knowledge about what to do in various situations in the world. Once chosen from the plan library, a plan is instantiated on the timeline, a constrained network of cognitive actions - the planner's cognitive agenda. The cognitive scheduler maintains the timeline, and is responsible for deciding which available action to execute. Communication with other agents is handled by the cognitive component; message processing is a cognitive activity, requiring mental effort. Some of these cognitive subcomponents are discussed in more detail below.

The cognitive timeline is a constrained network representing all of the cognitive activities that have been selected for future execution by the agent (except the simplest communication processing). Entries on the timeline are ordered temporally, with deadlines and coordination points represented explicitly; entries with no temporal constraints can be executed in arbitrary order. The cognitive scheduler maintains the timeline and controls the order of execution of timeline entries. Because entries are partially ordered, there are often strategic choices to be made about which action to execute next.

Timeline entries are either primitive cognitive actions or plans that expand into similar networks of entries that are added to the timeline. The execution method for a timeline entry is chosen when that entry is scheduled for execution. The expansion of a plan used as a plan step is done by the same mechanism - the entry representing the plan has an execution-method that causes an appropriate plan to be selected and its network of actions inserted into the timeline after the current action. By delaying the choice of execution-methods, the cognitive component can reason about its own use of cognitive time as the plan proceeds and choose execution-methods appropriate to real-time constraints.

Planning knowledge is stored in skeletal form in each agent's plan library. A plan is a partially ordered set of cognitive activities, where the ordering represents temporal constraints. Plans are indexed by environmental characteristics which make them applicable. When a plan is retrieved from the plan library, it is put on the timeline.

All agents in the Phoenix testbed have the same architecture. The generality of the agent architecture makes it possible to experiment with a number of issues, such as the authority relationships among agents, and various levels of functional capabilities within agents themselves. Another reason that the Phoenix agent architecture facilitates the design and construction of agents is that the Phoenix discrete event simulator is set up to handle agent processes, so the user need never think about how agent activities are interleaved and coordinated with other activities in the environment. The discrete event simulator automatically ensures that all activities get a "fair share" of time. For example, one can define a dozen agents and a scenario with a dozen fires, and never worry about how time is allocated to agents and fires.

Real time in Phoenix. In any simulation, there are two types of time to manage. One is cpu time and the other is simulation time. CPU time refers to the length of time that processes run on a processor. Simulation time refers to the "time of day" in the simulated environment. The Phoenix discrete event simulator kernel (Fig. 15.c) ensures that simulated parallel processes all begin or end at roughly the same simulation time, irrespective of the

cpu time they consume. For example, imagine it is now 12:00:00 in the simulated world, and an agent is about to begin planning. After one cpu second, simulation time for the agent is 12:05:00. The agent's adversary, in this case a forest fire, is "owed" five minutes of simulation time. It may take 7 cpu seconds to calculate the effects of five minutes of fire. Moreover, simulation time is still 12:05:00, because the agent and the fire are simulated parallel processes. So after eight cpu seconds (one for the planner and seven for the fire), we have simulated five minutes of planning and five minutes of fire, and both processes are paused at 12:05:00.

Note that a human observer keeps yet another type of time, which we call the observer's time. The planning and fire processes that took five minutes in simulation time took only about eight seconds (the eight cpu seconds plus some overhead) in the observer's time.

The relationship between simulation time and cpu time differs, depending on what kinds of processes the PHOENIX discrete event simulator kernel is running. Figure 16 illustrates these relationships.

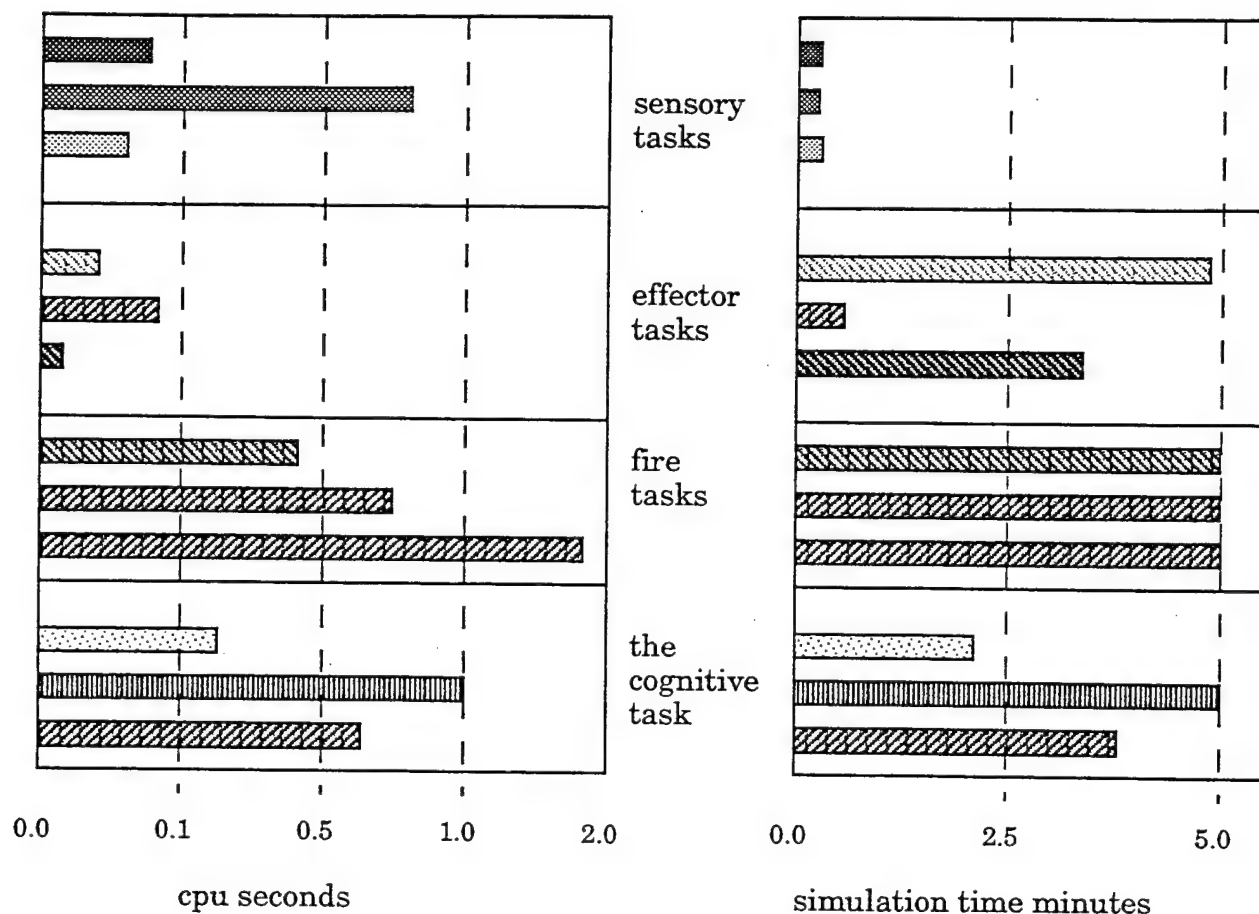


Figure 16. The relationships between cpu time and simulation time. Sensory tasks can, in principle, take an arbitrary amount of cpu time, but are instantaneous in simulation time. Effector tasks generally take little cpu time, and take an arbitrary amount of simulation time. Fire tasks take cpu time proportional to the size of the fire, and always take exactly five minutes of simulation time. CPU time and simulation time are strictly proportional for the cognitive task. Currently, one second of cpu time equals five minutes of simulation time. This is adjusted by the real-time knob.

Phoenix supports dozens of simultaneous processes, not just a single agent and a single fire, but it is easiest to understand what we mean by real time in this simple, two-process scenario. One process, the agent, is attempting to contain the other process, the fire. The term "real-time" refers to the relationship between the cpu time of the agent and the simulation time of the fire, irrespective of the cpu time the fire requires, and independent of the observer's time. It matters only that forest fires don't move too quickly (in simulation time) for a planner to decide how to control them (using cpu time). The definition of "real-time" has nothing to do with how much cpu time is required to calculate the movement of the fires, nor how quick or slow these processes appear to the observer. (For pragmatic reasons, however, Phoenix runs fire-fighting scenarios in minutes instead of days.)

More precisely, imagine that these processes, the planner and the fire, each have access to a clock that measures simulation time. Because these are simulated parallel processes, the planner and the fire are always allocated the same amount of simulated time. So when both have run for their allocated times

$$\text{elapsed-simulation-time(planner)} = \text{elapsed-simulation-time(fire)}$$

as measured by the simulation time clock. Although these processes could take arbitrary amounts of cpu time, it is advantageous to impose a strict relationship between cpu time and the simulation time of the planner and the fire. Thus,

$$\text{elapsed-simulation-time(planner)} = k * \text{cpu-time(planner)}$$

and, from the previous expression,

$$\text{elapsed-simulation-time(fire)} = k * \text{cpu-time(planner)}$$

The advantage of this proportionality is that we now have a way to exert time pressure on the planner. The real-time knob is the device that exerts pressure, simply by increasing k . Clearly, we can change k without changing the amount of cpu time allocated to the planner, and when this happens, the net effect is to increase the amount of simulation time allocated to the fire. Because of the strict proportionality between simulation time and cpu time for the planner, the indirect effect of increasing k is to reduce the amount of simulation time allocated to the planner, relative to the simulation time allocated to the fire. That is, to increase time pressure on the planner. Currently $k = 300$, which means that one second of cpu time for the planner is matched by five minutes of simulation time for the fire. If we increase k to 600, then the fire is allowed to burn for 10 minutes for every cpu second of planning time.

We reiterate that although this illustration is based on just two processes, Phoenix can manage dozens. Moreover, the principles are general: In a discrete event simulation, one

process or set of processes P will be "set against" another set F . The real-time problem is to ensure that P acts quickly enough to manage, or contain, or control F . Thus, P must use its cpu time efficiently, relative to F 's rate of change in simulation time. Furthermore, if P and F are simulated parallel processes, then elapsed simulation time for P must equal elapsed simulation time for F . This implies a strict proportionality between cpu time and simulation time for processes in P , but does not constrain the amount of cpu time required by processes in F .

6 Improvements to the PHOENIX Architecture

The PHOENIX architecture is intended to integrate multiple planning methods and decide in real-time which is most appropriate to use. The previous section described one kind of integration, between reactive and reflective planning. In this section we consider another kind of integration, specifically integration between agents. (These agents may, of course, adopt different planning methods.) In the operational planning domain, in particular, we have to consider integration between agents at different levels of a hierarchical command structure, as well as integration between semi-independent agents at the same level of a hierarchical structure. The concept of *envelopes* described in detail below, provides a mechanism for both kinds of integration. *Agent envelopes* are a mechanism to integrate the activities of agents at different levels of a hierarchical structure, while *plan envelopes* integrate agents at the same level of a structure under the jurisdiction of an agent at a higher level.

In the following discussion, we assume a two-level, hierarchical command structure. An agent at the higher level (call it P for planner) generates a plan and gives it to agents at the lower level to execute (call them A_1, A_2, \dots, A_n , for agents). In the fire-fighting domain, we expect P to be a reflective planner and A_i to be reactive planners. We discuss other arrangements in Section 8. We also assume that A_i is autonomous to some degree, and that P and A_i are not in continuous communication. Thus, P is uncertain about the exact location of A_i and its short-term (or tactical) activities, but P knows A_i 's longer term (or strategic) aims because P itself directed A_i to achieve them. Given these assumptions, which seem representative of the relationship between levels of command in operational planning, envelopes help us to answer some open questions:

- When should P and A_i communicate, and what should they say to each other?
- What information does P need to provide A_i besides goals?
- How can P integrate the activities of subordinate agents A_i and A_j ?

6.1 Envelopes

Plan monitoring in PHOENIX is accomplished with dynamic data structures called envelopes, which appear to have much in common with Standard Operating Procedures at some echelons of the Army Command and Control hierarchy (FM71-2, FM71-3). However, the similarity of these structures, envelopes and SOPs at *Corps* level, is uncertain. Nonetheless, it is interesting that these structures have been independently developed in the Army Command and Control hierarchy and the PHOENIX system, and we take it as evidence of the basic similarities between their operating environments.

An agent moves through a multidimensional space to achieve its goals. It moves through time and over physical distances, through areas with different physical characteristics, and through a space of information, which varies in quality and quantity. We call this space the agent's *environment*. The agent also moves through one or more spaces bounded by failure or other important events. These spaces are called *envelopes*. The concept is easier to illustrate than it is to define. Imagine you have one hour to reach a point five miles away, and your maximum speed is 5 miles per hour. If you are late, by even a moment, you fail. Clearly, to achieve your goal you must travel at unabated maximum speed for the entire distance. In this case, you *maintain your envelope* as long as your speed is 5mph, and you *lose* or *violate your envelope* if it drops below 5mph.

Consider another, less stressful example: You have one hour to travel five miles, as before, but your maximum speed is 10mph. You start slowly: your average speed is just 3mph. After 40 minutes you have travelled just two miles, and you have just 20 minutes to travel the other three. This is possible: If you travel at maximum speed (10mph), you will achieve your goal with about a minute to spare. On the other hand, if you continue to travel 3mph for another 171 seconds, you will fail—you will not be able to cover the prescribed five miles in one hour. In this example, you begin well within your envelope, but because your average speed is less than 5mph, you find yourself moving toward it. The rate at which you approach your envelope depends on your average speed: If it is 3mph, then it will take 42 minutes and 51 seconds to violate the envelope; if your average speed is only 2mph, then it will take just 37.5 minutes to violate your envelope. In contrast, if your average speed is greater than 5mph, then you will move away from your envelope. Figure 17 shows the progress of an agent that moves faster than 5mph until time 2, then slows down to less than 5mph. The shaded area shows the distance to the agent's envelope. Shortly after time 4, the agent achieves its goal with a little time to spare, that is, within its envelope.

Let us consider one more example before discussing the utility of envelopes in planning. Figure 18 shows a planner violate and then regain its envelope. Clearly, this is not a *failure envelope* of the kind we discussed above, because by definition a failure envelope (and thus one's goal) cannot be regained, even by maximum effort. The envelope in Figure 18 might be called a *warning envelope*. It represents a situation that is recoverable, but only by extraordinary action. Typically, there is just one failure envelope but many possible warning envelopes. To continue the previous example, you would violate a warning envelope if your average speed

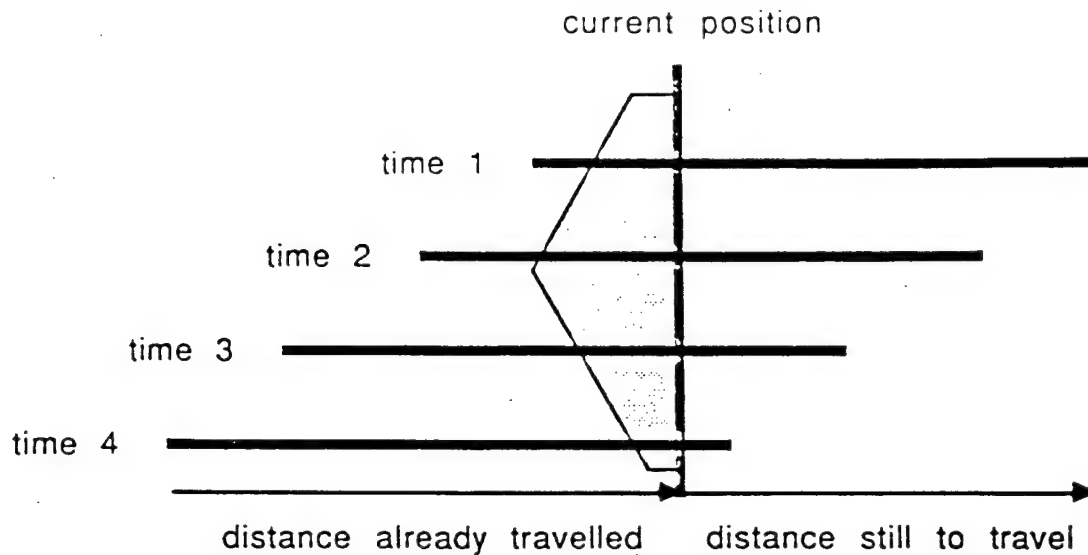


Figure 17: Planner approaching its failure envelope.

drops below 5mph, because this is the speed you must maintain to achieve your goal. Violating this envelope says "You can still achieve your goal, but only by doing better than you have up to this point."

6.2 The Utility of Envelopes.

The characteristics of envelopes that make them salient for planning in real-time, dynamic, uncertain domains are these:

- A planner can represent the progress of its plan by transitions within the plan's envelope. In Figure 17, the plan is "going smoothly," building up "slack" until time 2, at which point progress slows, and the slack gets used up. Progress, failures and potential failures are clearly seen from one's position with respect to envelopes, whereas this information is not apparent from one's position in the environment.
- Just as a planner can project how its actions will propel it through its environment, so it can project how these actions will move it with respect to its envelope. For example, in Figure 17 the planner can project, at time 1, how far ahead of schedule it will achieve its goal, given the assumption that the rate at which it is moving in the environment remains constant, and thus the rate at which it moves away from its envelope remains constant. Figure 19 shows how this might work with a different kind of envelope. At the top of the

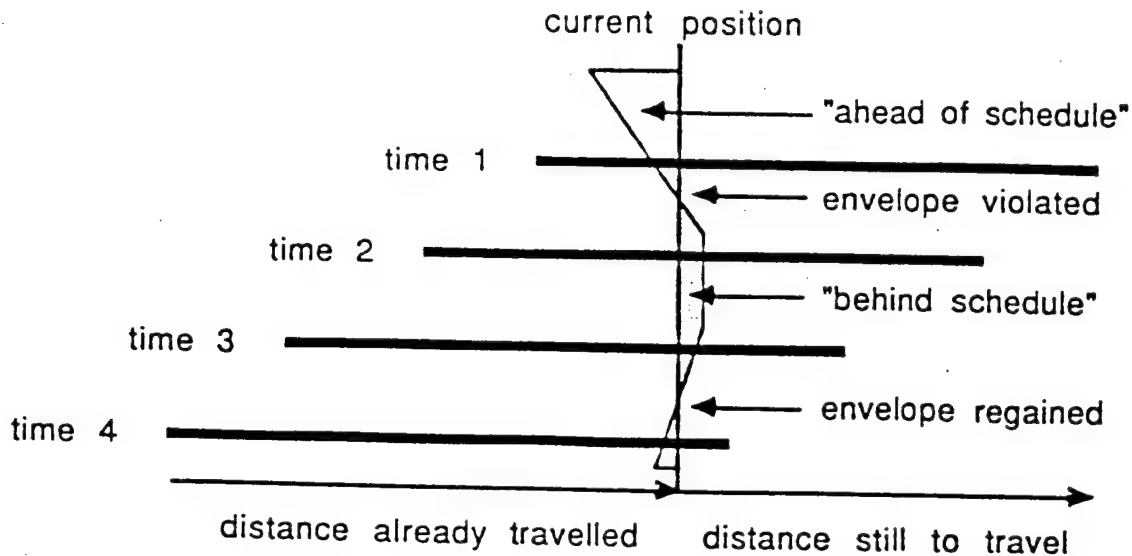


Figure 18: Planner approaching, violating, then regaining its warning envelope.

Figure, we see the progress of two bulldozers in containing a fire. One line in the graph below shows an envelope representing desired progress—the planned proportion of the fire that bulldozers have contained. The other line shows the proportion actually contained. Initially, the bulldozers are outside the envelope of the plan, because the proportion of area they contain is less than planned. However, the rate at which they are increasing this proportion is higher than planned, so we can project that they will eventually regain their envelope, and even when this will occur.

6.3 How Envelopes Integrate Projection and Reaction.

Here is how we propose envelopes should integrate these two styles of planning, and specifically how they integrate agents at different levels of a command hierarchy: A projective planner P formulates a goal and corresponding envelope by reflective planning, and gives them to a reactive planner A_i with the following instructions: "Here is the goal I want you to achieve. I don't care how you do it, and I don't want to hear from you unless you achieve the goal or violate the envelope." A_i then works independently, not monitored by P . If A_i is a bulldozer, it figures out where to go, how to avoid obstacles, and how to keep clear of the fire, until its goal is achieved or its envelope violated. Meanwhile, P is free to think about other agents, other goals, or to replan if necessary. Imagine that P gave A_i a warning envelope, and eventually A_i reports that it is violated. P can now look at A_i 's progress within its envelope. By projection

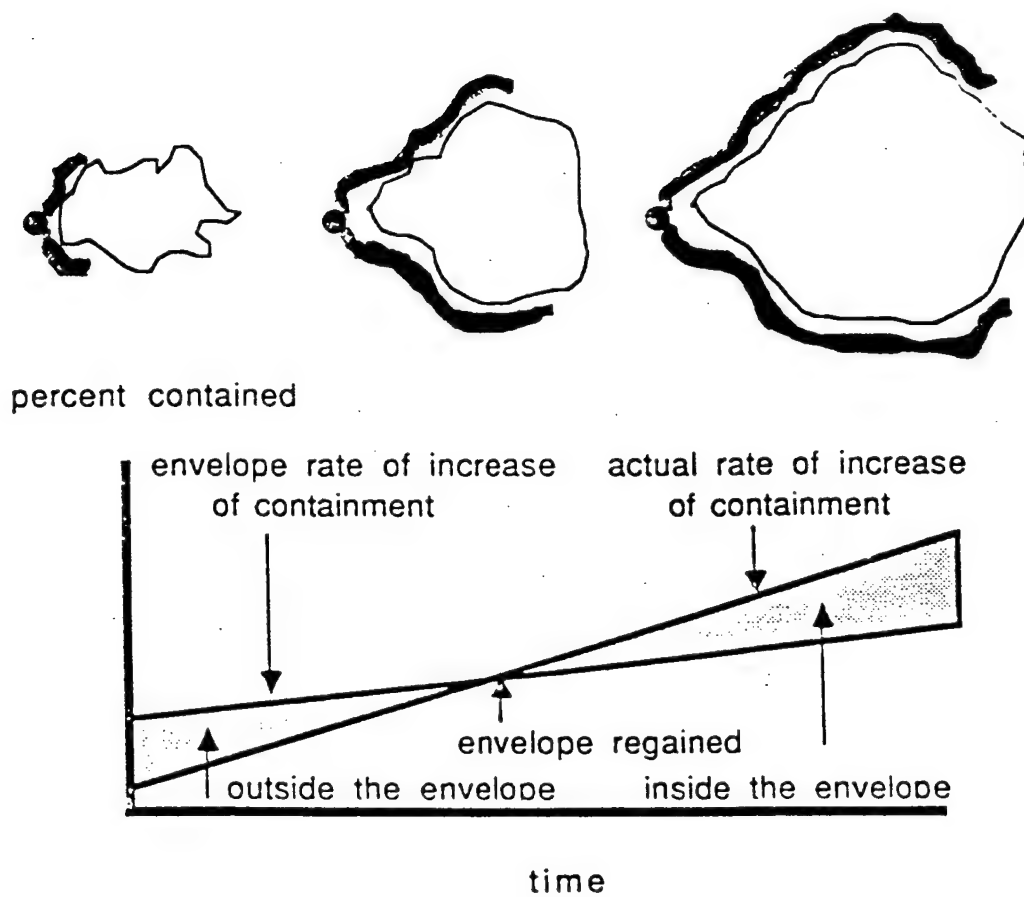


Figure 19: Planner begins outside the plan envelope, then regains it and continues to improve with respect to it.

P can see how late A_i will be in achieving its goal. How far is A_i from its goal? If it is nearby, the delay might be acceptable. But if A_i still has a long way to go, then it will violate its failure envelope relatively soon. This tells P that it should start formulating an alternative goal for A_i , and approximately when it should redirect A_i , assuming progress doesn't improve.

Envelopes grant agents a kind of autonomy, and grant planners the opportunity to ignore their agents until envelopes are violated. Once a planner directs an agent to achieve a goal, the agent can do what it likes as long until it violates an envelope, and the planner needn't monitor each of its agents actions (or each variation in the environment) unless an envelope is violated. Envelopes allow planners to use their computational resources efficiently—essential in real-time tasks.

6.4 Agent Envelopes and Plan Envelopes.

We distinguish between the envelopes of individual reactive agents and the envelopes of plans, which involve multiple agents. In Figure 19, for example, we show an envelope for a plan to contain a fire. The plan involves two bulldozers moving down the flanks of a fire and "pinching off" the front. Each bulldozer is a reactive agent with a goal and an envelope, but in addition, the entire plan has an envelope that represents the proportion of the fire the planner intended to be contained over time. One can imagine an individual bulldozer violating its envelope (e.g., cutting much less fire line than expected) but the plan as a whole staying within its envelope because the other bulldozer works more quickly.

One interesting research question is, how are plan envelopes generated? Is there a way to merge the envelopes of individual agents to get a joint envelope that represents their mutual progress? We believe it can be done with constraint satisfaction techniques, though we don't know how to do it yet.

Joint envelopes are needed to represent the effects of dynamic changes in the environment. Imagine that a planner has projected the progress of the fire assuming that the wind speed will remain constant, and it gives a bulldozer a destination and an envelope based on this projection. The bulldozer starts work and is well within its envelope when, unexpectedly, the wind speed increases. Now, if the bulldozer remains in its original envelope it might get burned, because the fire is moving more quickly than anticipated. The problem is that from the bulldozer's perspective, it is doing fine. It is within its envelope, and it doesn't know that this envelope is obsolete. Indeed, how would the planner itself realize this? One answer is ugly, the other elegant. The ugly answer is that the planner maintains dependencies between parameters and envelopes, so whenever a parameter changes the planner must update all the relevant envelopes. This is ugly because it isn't parsimonious and also because the planner must attend to all changes in the environment to see whether they have significant consequences. What we need is a way for the planner to ignore changes that are not significant. Envelopes define significance: The elegant solution is for the planner to construct a joint envelope for the bulldozer and the wind speed, so that if the bulldozer slows down, or the wind speed increases, or both, then the

bulldozer moves toward its envelope. If the wind speed continues to be high, the bulldozer will continue toward its envelope. If the wind speed drops, the bulldozer will move away from its envelope. Only when the bulldozer violates its envelope, due to continued high wind, or slow progress, or both, need the planner be notified. This means that the planner can ignore almost all changes in the environment, attending only to those changes that actually violate envelopes.

7 Envelopes in Operational Planning

In this section, we develop two examples of envelopes in operational planning and monitoring.

Any instruction, including those inside envelopes, can have an associated envelope. Thus, envelopes can be nested. Here's an example of an envelope associated with the instruction to defend an area:

Measure of progress: The total area under your control as a function of time (e.g., measured in hectares, at one hour intervals)

Thresholds:

Warning 1: Progress (change in hectares controlled over time) is negative (first derivative).

Warning 1 Instructions: If progress has been negative for two consecutive time periods, then project how long (in hours) it will be before Warning 3 or Warning 4 (whichever comes first), given the current progress, and let me know.²⁴

Warning 2: Negative progress is accelerating (second derivative).

Warning 2 Instructions: Project how long (in hours) it will be before Warning 3 or 4 (whichever come first), given the current rate of change of progress, and let me know.

Warning 3: The rate at which you are losing territory is N hectares/hour (first derivative = N hectares/hour).

Warning 3 Instructions: Let me know immediately.

Warning 4: If you continue at the current rate, you will lose something important (e.g., M hectares, or a particular bridge) in T minutes.

Warning 4 Instructions: Let me know immediately. Consolidate your resources to increase the amount of time you can hold the bridge, even if it requires giving up something else. Construct your own envelope for this.

Failure 1: You will not be able to maintain an area, or keep a bridge, or whatever; and you expect the area or object to fall in T minutes.

Failure 1 Instructions: Let me know immediately. Extricate yourself as quickly as possible.²⁵

²⁴We wait for two consecutive time periods because we don't want to overreact to ebb and flow; we want to be sure that progress is *really* negative. But note that if the rate of change of negative progress is increasing, we don't wait, because we couldn't know the rate of change was increasing without waiting at least three time periods.

²⁵The idea is that once it becomes clear that an agent will fail, there's no point devoting any more resources to trying to achieve success.

Failure 2: You have just lost M hectares, or a bridge, or whatever.

Failure 2: Let me know immediately.

Warning 5: Progress is positive.

Warning 5 Instructions: If positive progress continues for two time periods, then let me know immediately, and continue doing what you are doing within the following envelope:

Measure of progress: The total area under your control as a function of time (i.e., the same measure of progress as we have for the superordinate envelope).

Thresholds:

Warning 5.1: The rate at which you are gaining territory is increasing.

Warning 5.1 Instructions: Let me know immediately.

Warning 5.2: The rate at which you are gaining territory is M hectares/hour.

Warning 5.2 Instructions: Let me know immediately.²⁶

Warning 5.3: You now control P hectares.

Warning 5.3 Instructions: Hold that area but don't extend yourself any further. Let me know immediately.

Warning 6: Positive progress is accelerating.

...

...

This is about the simplest envelope one can construct. It involves a single agent, a single instruction—to defend an area—and a single measure of progress. We can easily make the envelope more complex by increasing the number of agents, instructions, and measures of progress. We will return to this point in a moment, but first we'll point out some of the salient characteristics of envelopes:

Envelopes are a medium of communication. It appears that communication bandwidths in battle are low, so messages have to be concise. An envelope is a message from a superordinate agent to a subordinate one, and it specifies precisely what messages the subordinate agent should send back to the superordinate, and under what conditions.

Envelopes specify how to characterize the dynamic behavior of an agent. Envelopes say which of the many possible measures of progress are germane, and so characterize dynamic behavior "in a nutshell." This is important because entire envelopes may be communicated when a force is transferred from one command to another, or when a commander needs to be briefed on the progress in an area. The

²⁶The point is that we don't want a force to overextend itself, so even if it is gaining territory we aren't necessarily happy; we need to be warned if it is increasing the rate at which it is gaining territory, or if it is gaining territory at a particular rate that is too fast.

analogy here is to a patient's chart. A physician halfway around the world can get a pretty good picture of a patient by relating the progress of a dozen or more vital signs. Note, however, that neither a physician nor a commander can get a good picture if they are told only what is currently happening. They need to know the history—how the situation has evolved. Since envelopes keep track of progress, they are a good medium for summarizing this information.

Envelopes as characterized above are pretty simple. I will now consider some more elaborate examples that extend the concept.

7.1 Agent Envelopes and Multi-agent Plan Envelopes

As we pointed out in Section 6, envelopes often measure the progress of multiple interacting agents. The interaction can be cooperative or competitive; forces can support or attack each other. We consider these cases in turn.

Dynamic Defense. Imagine the situation in which two forces are falling back toward a river. The exact orders are, perhaps, to hold the sector that includes the river and counterattack. By mechanisms like those described in Powell and Schmidt's paper, these orders are elaborated and operationalized to something like this:

Maintain a spatial relationship between the two forces such that neither can be isolated by enemy forces; maintain a division in reserve (to counterattack); be in a particular configuration to cross the river.

(Indeed, it is easy to see that these require still more operationalization, but I'll leave it at that.)

Four envelopes seem important here. One measures the progress of both forces holding the sector. It looks like the envelope above, except that it requires information from both forces. The second envelope measures the spatial relationship between the two forces. The third is very similar: it measures whether the forces are making progress toward rearranging themselves into the configuration they require to cross the river. The fourth envelope measures the number of troops committed to battle, and is violated if any members of the last division are committed.

It isn't difficult to write functions for these envelopes; each measures something that is pretty straightforward and easy to assess. But there are questions about who constructs these envelopes and who supplies the information for them. Consider the progress of forces holding a sector. Should one of the two forces maintain this envelope or should it be done at the command post? In the first case, one of the forces must tell the other one how it is doing; in the second,

each force must tell the command post how it is doing. The same point holds for the next envelope: To determine whether two forces are in the right spatial relationship over time, they must either communicate amongst themselves, or with their common command post.

Which should it be? Should envelopes for subordinates be maintained by the subordinates themselves, or by the superordinate? The answer depends on several factors, which we will discuss in detail following the next example.

Envelopment We now consider a case of competitive interaction between forces, specifically, competition with enemy forces and cooperation among one's own forces. The example is a familiar tactic which consists of a purposeful delay used to shape a penetration that sets up the conditions for a double envelopment. In this paper we will refer to this tactic as an envelopment. One's own forces fall back in a parabolic line, drawing the enemy forces into a pocket, where they are surrounded. Imagine an envelopment is underway, and we are monitoring its development to see whether it is working the way we want. What needs to be monitored? The following is a partial list:

- Are our own forces forming a parabolic "pocket" as they fall back?
- Are our own forces distributed in the pocket as we wish?
- Are the enemy forces moving into the pocket as we wish? Or are they moving too far in, and threatening to break through? Are they failing to enter the pocket, not engaging us, slipping away? Are they holding back for some other reason, perhaps to await reinforcements? Are they concentrating on the flank of the pocket, to envelope our flank and mount a counterattack?
- Is our envelopment proceeding as required by our more global strategy?
- Does the enemy have reinforcements within range, or artillery in range, to overwhelm us in the envelopment area?

All these things can be monitored with envelopes. Before we describe how to do it, we need to characterize the envelopment tactic more abstractly and precisely, as shown in Figure 20.

The first pane in Figure 20 shows an envelopment in progress, with own forces forming a parabola, roughly symmetric around centerline c . The point d is a coordination point at which 1) the center of gravity of the enemy forces (*cog*) and 2) the onset of the envelopment must intersect in time. In other words, for the envelopment to be successful, *cog* must be at d , and the flanks begin to close, simultaneously. If the flanks begin to close too late, or *cog* isn't at d when they begin, the envelopment may fail. In practice, d is a real point on the battlefield, but it is unknown until shortly before the envelopment begins. This is a pretty common occurrence in real-time planning, where we often plan to achieve states that satisfy particular predicates, without knowing exactly which states they are. In this case, we want *cog* to be at a place that satisfies at least these predicates:

- *cog* is on the centerline *c*, or is not “too far” from it, where too far is defined in terms of the distribution of own forces in the pocket, and the exact shape of the pocket.
- *cog* is too far into the pocket to escape once the flanks begin to close (assuming that the flanks close without interruption, at the expected rate).
- *cog* is not so far into the pocket that the forces at the flanks risk firing on own forces at the rear of the pocket.

These, then, are some of the components of what we mean by “progress” in an envelopment. They partly define an envelope, which we might call the **relative-position-envelope**, and which measures whether the enemy forces are “moving into position,” relative to ours, for us to execute the envelopment successfully. With some additional machinery, described below, we can monitor progress on the three measures above to see how the envelopment is developing. Without knowing exactly where the enemy forces will be when the envelopment begins (*d*), we can nevertheless ask whether they are moving toward such a point or away from it.

To do this, we have to define a few more measures to operationalize the three just described. Consider the first one: *cog* is on the centerline *c*, or is not “too far” from it, where too far is defined in terms of the distribution of own forces in the pocket, and the exact shape of the pocket. One easy operationalization of this involves plotting the trajectory of *cog* relative to the trajectory of the *focus* of the parabola formed by our own forces, abbreviated *f*. This is shown in the center pane of Figure 20. In part (b), the distance between *f* and *cog* is decreasing over time, and in part (c) it is increasing. Clearly, in the former, the enemy forces are moving into a position advantageous to us, and in the latter they are not. We could easily construct more accurate measures, but this simple one—the first derivative of the distance between *f* and *cog*—would probably suffice. For later reference, we’ll call this measure the *degree of convergence* and abbreviate it *doc*.

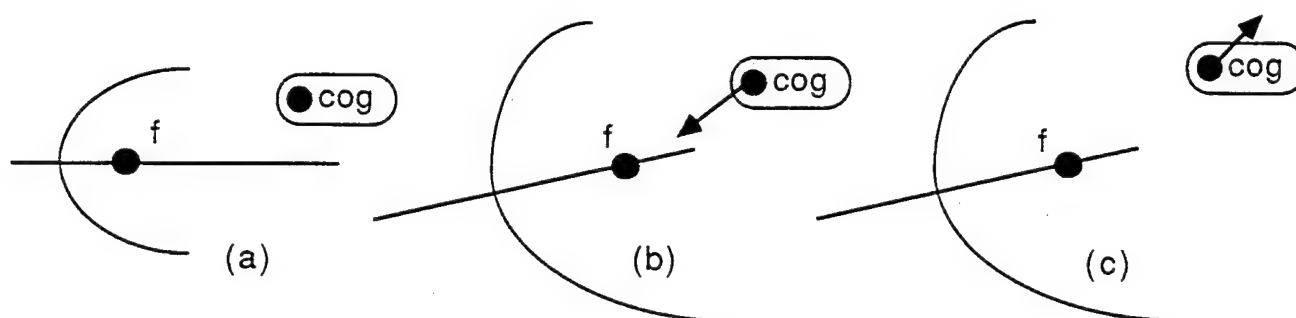
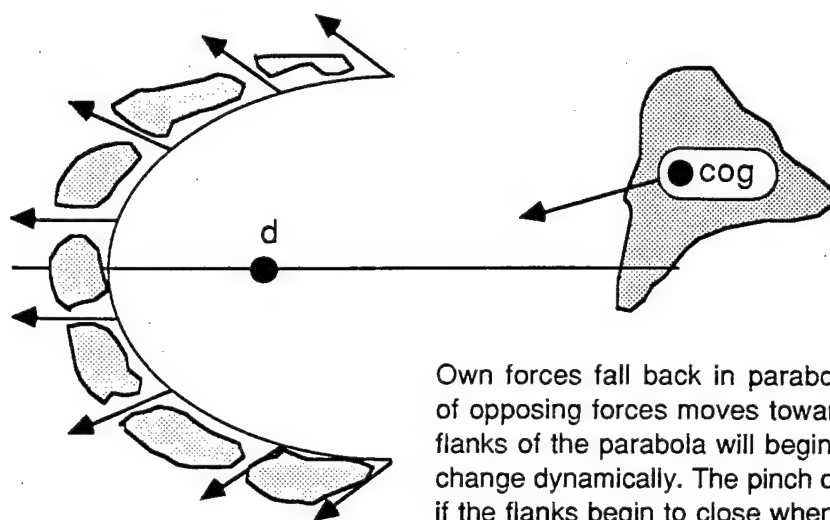
Now consider the other criteria for progress in the **relative-position-envelope**: *cog* must be far enough inside the pocket to prevent escape, and not so far that the flanks fire on their own forces at the back of the pocket. The former measure depends on the *cog*’s feasible rate of movement. Fast enemy forces must be deeper in the pocket than slow forces. Once again, we can define a measure to determine whether we are moving towards a successful envelopment or away from one. This measure should probably include more parameters than the last one:

l_1 – the distance between the endpoints of the flanks on the line denoted *l* in Figure 20

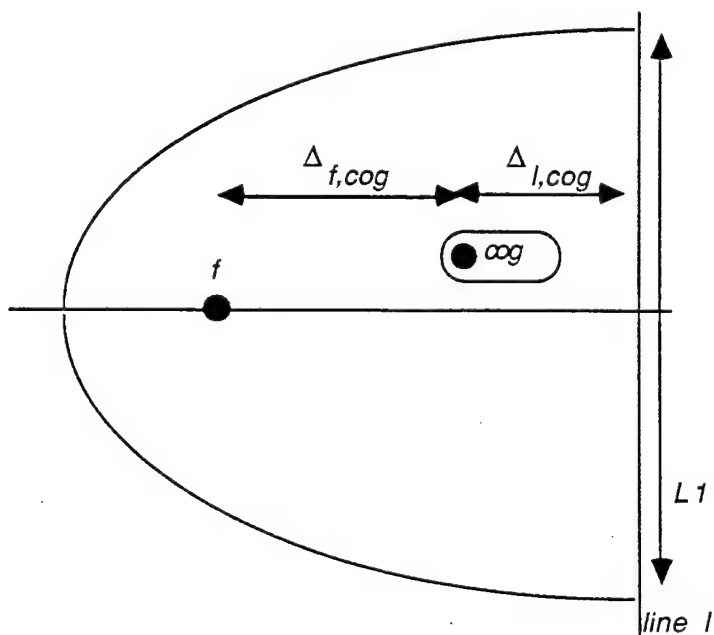
v_{own} – the maximum velocity of own forces (specifically, those on the flanks)

$\Delta_{l,cog}$ – the distance between *cog* and the nearest point on line *l*

v_{cog} – the maximum velocity of *cog* (more specifically, the maximum velocity out of the pocket.)



Relative to the configuration in (a), the progress of cog toward the focus of the parabola, f, is increasing in (b) and decreasing in (c)



Components of a more complex envelope.

- $\Delta_{l,cog}$ -- distance to travel before cog "escapes" the pinch out
- $\Delta_{f,cog}$ -- distance to travel before the pinch out can begin with high probability of success
- $L1$ -- distance the flanks have to close

With these parameters, we can define a measure to operationalize “far enough into the pocket that they can’t escape.” Clearly, this measure depends on the rate at which we can close the pocket, determined by l_1 and v_{own} , and the ability of the enemy to retreat out of the pocket, determined by $\Delta_{l,cog}$ and v_{cog} . As before, we can make this measure arbitrarily complicated by adding more and more parameters, but these will probably suffice.

Lastly, we consider the criterion that we shouldn’t fire on our own forces. This is a simple measure that depends on the angle of the parabola and the position of *cog* within it. Specifically, we are concerned with the position of *cog* on the axis *c*. Let’s define a measure called $depth_c$ as follows: $depth_c = \Delta_{l,cog} / \Delta_{f,cog}$ (see the third pane of Fig. 20). Thus, if $depth_c > 1$, *cog* is “more than halfway” into the pocket, and, conversely, if $depth_c < 1$ then *cog* is relatively near the exit of the pocket. Clearly, we want $depth_c$ to be large enough that *cog* can’t escape, but small enough that we don’t fire on our own forces.

Two points about this analysis must be stressed. First, all these measures are *dynamic*, meaning that their values typically change over time. Consider degree of convergence, the first measure we described. If *doc* is negative it means that the distance between *f* and *cog* is decreasing, and *cog* is moving into the pocket. If $doc > 0$ it means *cog* is moving away from *f*, and perhaps out of the pocket. *doc* itself can be constant, but is more likely to be dynamic, given that *cog* and *f* are simultaneously changing (because both enemy and own forces are moving).

Second, the conditions for a tactic, such as an envelopment, are typically specified by *several* measures like *doc* or $depth_c$, not just one. For example, the **relative-position-envelope** involves *doc* and $depth_c$, which tells us both whether the enemy can escape, and whether we risk firing on our own forces. There is therefore a *probable success envelope* of any plan, such as an envelopment, in which it is likely to succeed (in fact, there are numerous probable success envelopes for a given plan, corresponding not only to the conditions under which success is probable, but also to different probability levels). The probable success envelope is a subspace of the space defined by the parameters described earlier: *doc*, $depth_c$, *cog*, *f*, l_1 , v_{own} , $\Delta_{l,cog}$, $\Delta_{f,cog}$, v_{cog} and the angle of the parabola (approximated by l_1 and the length of *c*). It is up to the planner to specify the envelope for a plan. Because it is difficult to specify a complete function relating all these parameters and measures, we would expect the envelope to be specified in *tabular* form [8], perhaps in a set of rules, based on subsets of the parameters and measures. For example, the planner may stipulate that the envelopment is likely to succeed only when these conjunctive conditions hold:

- $doc < 0$ (the enemy is moving into the pocket)
- $l_1 > 2 * length(c)$ (flanks are unlikely to fire on center)
- $v_{own} > v_{cog}$ (own forces can outrun enemy)

- $.6 < depth_c < 1.0$ (*cog* is at least partway into the pocket and at most halfway in.)

In fact, the planner can stipulate numerous rules of this kind for when the envelopment is likely to succeed. Each defines an probable success envelope in which the envelopment is believed, to some degree, likely to succeed.

As we noted, all these parameters and measures are dynamic, which means they change value over time. Moreover, the planner is likely to start out *outside* its probable success envelope; for example, the parabolic shape of our own forces must evolve before the envelopment is possible, and until that time, the plan is not within a probable success envelope. These observations have two consequences. First, it is difficult to predict exactly *when* the planner will enter a probable success envelope. Second, the planner may *never* enter a probable success envelope. For example, our own forces may fall back, as desired, and *cog* of the opposing forces may fall into the pocket, as desired, but the *cog* may always be too deep in the pocket for us to risk the envelopment. At some point, we may have to declare the envelopment a failure and try something else.

Note that probable success envelopes don't help here: they tell us only when success is likely, they don't tell us when failure occurs. Probable success and failure are not exhaustive alternatives; a plan can be outside a probable success envelope indefinitely, and may still eventually enter the envelope and succeed. But it is this characteristic of probable success envelopes that really requires failure envelopes. Clearly, our planner cannot go through life saying "Let's give it just a little longer, and perhaps conditions will change for the better." At some point, it has to realize that it is too late for conditions to change, and the plan has failed. Here's a simple example from the envelopment plan. Imagine, as described earlier, that own forces start to form a parabolic pocket, but *cog* of the opposing forces stays very close to the rear of the pocket. The longer this continues, the deeper the pocket, and the greater the danger of the flanks firing on own forces, and being unable to reinforce the rear of the pocket in case of a breakout attempt. Equipped only with probable success envelopes, the pocket may continue to deepen, in the hope that some parameters might take on values that would allow the plan to enter a probable success envelope. Failure envelopes are needed to say, essentially, the situation is not going to get better, so we'd better stop it right now.

8 Mapping the PHOENIX Architecture, with Envelopes, to the Operational Planning Problem

The purpose of this section is to discuss consider in detail the claim that the PHOENIX architecture is a good first attempt at an architecture for operational planning. We have structured the section as a series of questions and their answers. We want to stress that the claims in this section are preliminary.

Q: Are the fire and OP environments so different that PHOENIX is irrelevant?

A: Obviously we don't think so, but there are differences. In military domains, one must deal with a sentient adversary; whereas one's adversary in fire-fighting is unpredictable, dangerous, and dynamic, but hardly malicious. The similarities between the domains, however, are striking:

- Resources are allocated to control threats
- There are multiple threats and multiple resources
- This process is ongoing and dynamic
- There is considerable uncertainty about the threats and, to a lesser extent, about one's own resources
- All planning takes place in real time
- The environment is spatial and involves moving resources from one place to another, which takes time and communication
- Resources have different capabilities and are best used for different purposes
- Weather, terrain, and other environmental factors have a profound influence on the probability of success

Although the current fire fighting environment is small compared with the European theater, we think it unlikely that this limits the applicability of PHOENIX to operational planning.

Q: Is the structure of the planners too different?

A: The answer must account for several things:

- PHOENIX uses just two "levels of command," the central planner and its agents, whereas the military has many levels of command. Will the communication mechanisms for PHOENIX work for multi-level hierarchies? We believe the envelope idea can be extended to multiple levels.
- Communication among agents at a single level of the hierarchy. The idea of a plan envelope is that an agent at one level of a command hierarchy can construct a joint envelope for multiple agents under its control (this envelope can include environmental conditions and the actions of adversaries). Plan envelopes give us the data structures we need to manage multiple agents, but they do not address the possibility that these agents can communicate cooperatively among themselves to form a plan. They assume instead that the plan has been "handed down" from a higher level. This may be the correct assumption for operational planning. In any case, one project in our lab is to develop a truly distributed planner for the fire problem, based on envelopes.

- Decision-making seems to involve the negotiation of experts in many areas (although a single agent always makes the final decision). We can't model this negotiation process in PHOENIX.

Q: Do we need to simulate an engagement to handle either the static or dynamic aspects of the OP problem?

A: The static aspect requires simulation, or projection, at some level of abstraction. One cannot place resources on a map without considering, at some level of abstraction and to some time horizon, how an engagement would proceed. Though this seems difficult, two factors make it less so:

Telltale. It is sometimes unnecessary to project a plan into the future to see how well it works. This is because we can often "read the future" from aspects of the present. We call these aspects *telltale*s. One telltale from the fire domain is a "loop" in the fire; we know from experience that if crews or bulldozers enter a loop, they will get trapped if the loop closes. It is unnecessary to project all possible outcomes of crews and bulldozers entering loops. In fact, it is unnecessary to project at all. The simple recognition of a loop is enough to warn the planner to stay away. A telltale reduces the colossal combinatorics of projection—simulation—by replacing a projection problem with a recognition problem.

Telltale are numerous in operational planning. They are evident, for example, when a planner says "He won't come this way because there are too many rivers to cross." The rivers, like the loop in the fire, enable the planner to prune alternatives from the space of possible futures.

Abstraction. One lesson of AI planning research is that complex processes can be decomposed into "nearly independent" subprocesses or, alternatively, clustered into nearly-independent units. So, for example, military forces are not modelled in terms of individuals, but in terms of clusters of individuals that have particular properties, such as an artillery battalion. Clearly, it is easier to simulate an engagement at the level of battalions than at the level of individual troops. In general, the higher the level of abstraction, the easier the simulation, and the more "detail" is lost. Given this fundamental tradeoff, simulation for planning requires finding the right level of abstraction. A good planning architecture will permit planning at multiple levels of abstraction, so some aspects of the plan can be crudely sketched and simulated, while others—presumably more critical aspects—are dealt with in detail.

Fortunately, the hierarchical organization of military forces gives us some idea about the appropriate levels of abstraction. However, we find in operational planning the creation of ad hoc clusters, such as a covering force with an artillery backup, that are treated as a single object. These objects have components from different levels of abstraction (one may be a battalion and the other a handful of brigades), and they may have different characteristics, and serve different purposes. Nevertheless, they may be treated as a single

unit for the purposes of planning. All this suggests that while the power of abstraction is clearly essential, it isn't as clear how components of forces at different levels of abstraction are created.

The dynamic aspect of the OP problem also requires some simulation or projection. When allocating reinforcements, advancing or withdrawing forces, and otherwise managing one's resources, one needs some idea of how a plan will fare before attempting it. The problems with this approach were outlined earlier (Sec. 4). Chief among them is that all planning takes time, during which one's resources may be depleting rapidly, or during which opportunities may be lost. We handled this problem four ways in PHOENIX (and in the proposed envelope-based extension to PHOENIX):

Plan when there is time. In fire-fighting, the planner is frequently waiting for the fire-fighting objects to move from one place to another, or build a prescribed line, or report what they see. During this time, the planner considers alternative plans. Consequently, if a current plan fails, or begins to fail, the planner often has another one ready.

Plan in advance; contingency planning. An extreme form of the previous point is that plans should be worked out at some level of abstraction, with many contingencies considered, before an engagement begins. Clearly, the level of abstraction of contingency plans depends on how many and how unpredictable are the outcomes once the engagement begins. In the forest-fire problem, we maintain a library of skeleton plans that specify, at the right level of abstraction, how to deal with different kinds of fires. There's a plan for containing small fires in light winds, another for containing fires that threaten property, and so on.

Have dynamic control of the depth and level of abstraction of dynamic replanning. Since it is clearly impossible to anticipate *all* contingencies at an acceptable level of abstraction, the planner will have to deal with unexpected situations as they arise. This will require projection which, as we noted above, takes time. However, sometimes unexpected situations evolve slowly and sometimes very fast; so sometimes the planner can carefully consider how to respond, and sometimes it has to react immediately. Consequently, the planner requires dynamic control of how much projection it does. The more time, the more projection, all other things being equal. The time required for projection in turn depends on the depth of the projection (the number of moves and countermoves that are considered) and the level of abstraction at which the projection takes place. The PHOENIX architecture offers dynamic control of both these parameters.

Use envelopes to minimize interactions between the planner and its agents, and to project progress. Envelopes, as we said in Section 6, measure the progress of a planner's agents. They minimize the interaction between the planner and its agents—and thus the amount of time the planner must spend monitoring and replanning for its agents—by telling the agents, in effect “Do this, I don't care how, and I don't want to be bothered unless things are going especially well or especially badly, which I define as

follows..." This seems to be the nature of interaction between levels of command in the military hierarchy, so we expect envelope-based planning to be essential to the dynamic, engagement phase of the operational planning problem.

The other important contribution of envelopes is that because they measure progress, they enable the planner to project the progress of its agents. This gives the planner valuable warning of impending crises.

9 Conclusion

As of this writing, we have implemented envelopes, approximate processing, and replanning in Phoenix. All these abilities are rudimentary, however, and much more needs to be done. Yet, Phoenix is clearly an appropriate architecture for operational planning and monitoring. Over the course of days of simulated time, the Phoenix planner detects threats (forest fires), assesses its resources, plans appropriately, monitors the execution of the plans (despite message delays and other kinds of noise), and modifies its plans when the environment so dictates. It does all this in real time, that is, in a timely way given the rate at which the environment changes. In PHOENIX we have a laboratory in which to study the operational planning problem and the applicability of AI technology to it.

References

- [1] Philip E. Agre and David Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 268-272, Seattle, Washington, 1987. American Association for Artificial Intelligence.
- [2] James F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123-154, 1984.
- [3] Rodney A. Brooks. A robust layered control system for a mobile robot. A.I. Memo 864, Massachusetts Institute of Technology, 1985.
- [4] Carol A. Broverman and W. Bruce Croft. Reasoning about exceptions during plan execution. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 190-195, Seattle, Washington, 1987.
- [5] David Chapman and Philip E. Agre. Abstract reasoning as emergent from concrete activity. In M. P. Georgeff and A. L. Lansky, editors, *Reasoning About Actions and Plans, Proceedings of the 1986 Workshop at Timberline, Oregon*, pages 411-424, 1987.
- [6] Paul R. Cohen and David S. Day. The centrality of autonomous agents in theories of action under uncertainty. Ekl technical report, University of Massachusetts, January 1988. To appear in the *International Journal for Approximate Reasoning*.
- [7] Paul R. Cohen and Edward Feigenbaum. *The Handbook of Artificial Intelligence, Volume III*. William F. Kaufmann, Inc., Los Altos, CA, 1982.
- [8] Paul R. Cohen, Glenn Shafer, and Prakash P. Shenoy. Modifiable combining functions. *AI EDAM*, 1(1):47-85, 1987.
- [9] U.S. Army Command and General Staff College. *ST 101-9 The Command Estimate*. Ft. Leavenworth, Kansas, 1987.
- [10] Jim R. Dacus. Knowledge-based planning in advanced tactical aircraft. In *Proceedings of the DARPA Knowledge-Based Planning Workshop*, Austin, Texas, December 1987.
- [11] M. Daily, J. Harris, D. Deirsey, K. Olin, D. Payton, K. Reiser, J. Rosenblatt, D. Tseng, and V. Wong. Autonomous cross-country navigation with the ALV. In *Proceedings of the DARPA Knowledge-Based Planning Workshop*, Austin, Texas, December 1987.
- [12] Thomas Dean. Large-scale temporal data bases for planning in complex domains. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, 1987.
- [13] Thomas Dean. Planning, execution and control. In *Proceedings of the DARPA Knowledge-Based Planning Workshop*, Austin, Texas, December 1987.

- [14] Edmund H. Durfee. A unified approach to dynamic coordination: Planning actions and interactions in a distributed problem solving network. Ph.d. dissertation, University of Massachusetts, Amherst, Massachusetts, September 1987.
- [15] R. Fikes, P. Hart, and N. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3(4):251-288, 1972.
- [16] R. James Firby. An investigation into reactive planning in complex domains. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 202-206, Seattle, Washington, 1987.
- [17] R. James Firby and Steve Hanks. A simulator for mobile robot planning. In *Proceedings of the DARPA Knowledge-Based Planning Workshop*, Austin, Texas, December 1987.
- [18] Alan Garvey, Craig Cornelius, and Barbara Hayes-Roth. Computational costs versus benefits of control reasoning. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 110-115, Seattle, Washington, 1987.
- [19] Steve Hanks. Temporal reasoning about uncertain worlds. In *Third Workshop on Uncertainty in Artificial Intelligence*, pages 114-122. American Association for Artificial Intelligence, July 1987.
- [20] B. Hayes-Roth, A. Garvey, M.V. Johnson, and M. Hewett. A layered environment for reasoning about action. Technical Report KSL 86-38, Computer Science Department, Stanford University, April 1986.
- [21] Barbara Hayes-Roth. Dynamic control planning in adaptive intelligent systems. In *Proceedings of the DARPA Knowledge-Based Planning Workshop*, Austin, Texas, December 1987.
- [22] James A. Hendler and James C. Sanborn. A model of reaction for planning in dynamic environments. In *Proceedings of the DARPA Knowledge-Based Planning Workshop*, Austin, Texas, December 1987.
- [23] Martin Herman and James S. Albus. Real-time hierarchical planning for multiple mobile robots. In *Proceedings of the DARPA Knowledge-Based Planning Workshop*, Austin, Texas, December 1987.
- [24] Richard Korf. Real-time heuristic search: First results. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 133-138, Seattle, Washington, 1987.
- [25] Victor R. Lesser, Edmund H. Durfee, and Jasmina Pavlin. Approximate processing in real-time problem solving. *AI Magazine*, pages 49-61, Spring 1988.
- [26] Richard A. Luhrs and Anthony R. Nowicki. Real-time dynamic planning for autonomous vehicles. In *Proceedings of the DARPA Knowledge-Based Planning Workshop*, Austin, Texas, December 1987.

- [27] Drew McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6:101-155, 1982.
- [28] Drew V. McDermott. DARPA-sponsored planning research: Report and prospectus. YALE/CSD/RR 522, Yale University, Department of Computer Science, March 1987.
- [29] A. Newell and H.A. Simon. *Human Problem Solving*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1972.
- [30] HQ Department of the Army. *FM 101-5 Staff Organization and Operations*. Department of the Army, 1984.
- [31] William J. Pardee and Barbara Hayes-Roth. Intelligent real time control of material processing. Technical report, Rockwell International, Science Center, Palo Alto Laboratory, February 1987.
- [32] E. D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5(2):115-135, September 1974.
- [33] E.D. Sacerdoti. *A Structure for Plans and Behavior*. American Elsevier: New York, NY, 1975.
- [34] G. J. Sussman. *A Computer Model of Skill Acquisition*. American Elsevier, New York, NY, 1975.
- [35] William Swartout. Summary report on DARPA Santa Cruz workshop on planning. In *Proceedings of the DARPA Knowledge-Based Planning Workshop*, Austin, Texas, December 1987.
- [36] A. Tate. Generating project networks. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 888-893, Tbilisi, Georgia, USSR, 1977.
- [37] Austin Tate. INTERPLAN: A plan generation system that can deal with interactions between goals. Machine Intelligence Research Unit Memo MIP-1-109, University of Edinburgh, Edinburgh, December 1974.
- [38] Stephen Vere. Planning in time: Windows and durations for activities and goals. Technical report, Jet Propulsion Laboratory, Pasadena, California, 1981.
- [39] Richard Waldinger. Achieving several goals simultaneously. In E. W. Elcock and D. Michie, editors, *Machine Intelligence, Volume 8*. Halstead/Wiley, New York, NY, 1977.
- [40] D. H. D. Warren. WARPLAN: A system for generating plans. Department of Computational Logic Memo 76, University of Edinburgh, 1974.
- [41] Michael P. Wellman and David E. Heckerman. The role of calculi in uncertain reasoning. In *Third Workshop on Uncertainty in Artificial Intelligence*, pages 321-331. American Association for Artificial Intelligence, July 1987.
- [42] David E. Wilkins. Recovering from execution errors in SIPE. Technical report, 1985.